Penerapan Metode YOLO dan Tesseract-OCR untuk Pendataan Plat Nomor Kendaraan Bermotor Umum di Indonesia Menggunakan Raspberry Pi

Eric Tirtana
Program Studi Informatika
Fakultas Teknologi Industri
Universitas Kristen Petra
Jl. Siwalankerto 121-131
Surabaya 60236
Telp. (031)-2983455
erictirtana13@gmail.com

Kartika Gunadi
Program Studi Informatika
Fakultas Teknologi Industri
Universitas Kristen Petra
Jl. Siwalankerto 121-131
Surabaya 60236
Telp. (031)-2983455
kgunadi@petra.ac.id

Indar Sugiarto
Program Studi Teknik Elektro
Fakultas Teknologi Industri
Universitas Kristen Petra
Jl. Siwalankerto 121-131
Surabaya 60236
Telp. (031)-2983455
indi@petra.ac.id

ABSTRAK

Sistem parkir merupakan hal yang sering ditemui pada tempat-tempat umum. Sistem parkir biasanya memiliki sebuah program untuk mengenali dan membaca plat nomor. Seiring berkembangnya teknologi, program / sistem yang dapat mengenali dan membaca plat nomor secara otomatis sudah tersedia secara luas. Namun jika sebuah pihak ingin menggunakan sistem tersebut, terkadang dibutuhkan biaya yang cukup mahal.

Pada penelitian ini akan digunakan Raspberry Pi 4. Dengan menggunakan Raspberry Pi, diharapkan dapat mengurangi biaya yang harus dikeluarkan untuk dapat mencapai hasil seperti program yang sudah ada. Namun menggunakan Raspberry Pi juga terdapat kekurangan, yaitu spesifikasi hardware yang tidak sebagus komputer seperti biasanya. Pada penelitian ini akan digunakan metode YOLO (You Only Look Once) untuk mendeteksi plat nomor dan Tesseract-OCR untuk membaca karakter yang terdapat pada plat nomor tersebut.

Dari penelitian ini, dapat diambil kesimpulan bahwa program dapat menerapkan YOLO dan Tesseract-OCR untuk mendeteksi dan membaca plat nomor kendaraan bermotor umum dengan dijalankan pada Raspberry Pi 4. Untuk mendapatkan hasil yang optimal, gambar yang menjadi input dapat diambil pada siang hari, dengan menggunakan kamera berkualitas tinggi, dan untuk metode pre-processing hanya menggunakan metode-metode yang diperlukan saja.

Kata Kunci: plat nomor, YOLO, tesseract-OCR, Raspberry Pi

ABSTRACT

Parking system is a common thing to find in public places. Parking system usually comes with a program that enables to detect and read license plates. With the advancement of technology, there are many systems / programs that are able to automatically detect and read license plates, but they come with a costly price.

In this research, Raspberry Pi 4 will be used as the main platform. With the usage of Raspberry Pi, it is expected to reduce the cost needed to achieve the same output. However, by using Raspberry Pi, the hardware specifications are not as good as computer in general. In this research YOLO will be used to detect the license plate and Tesseract-OCR is used to read the characters on the license plate.

From this research, it can be concluded that program can implement YOLO and Tesseract-OCR to detect and read public transportation license plates while being run on Raspberry Pi 4. To get the optimal results, the input image needs to be taken at daytime, using high quality camera, and implement only the necessary pre-processing methods.

Keywords: *license plate, YOLO, tesseract-OCR, Raspberry Pi*

1. PENDAHULUAN

Kendaraan bermotor menjadi salah satu mode transportasi yang paling sering digunakan oleh masyarakat Indonesia. Berdasarkan data dari Badan Pusat Statistik, jumlah kendaraan bermotor pada tahun 2018 mencapai angka 146 juta. Di dalam undang-undang yang berlaku, sebuah kendaraan bermotor wajib menggunakan Tanda Nomor Kendaraan Bermotor, yang biasa disebut dengan plat nomor. Di Indonesia, terdapat 5 jenis plat nomor, salah satunya yaitu plat nomor kendaraan bermotor umum, dengan ciri khasnya memiliki dasar warna kuning dan huruf / angka berwarna hitam.

Di era dengan teknologi yang sudah maju seperti ini, program untuk pengenalan dan pembacaan plat nomor pun sudah tersedia secara luas. Bahkan, pada masa kini sudah terdapat beberapa perusahaan yang berfokus untuk menyediakan sistem parkir secara otomatis dengan menggunakan *software* untuk mengenali dan membaca plat nomor, serta menyediakan peralatan-peralatan yang diperlukan, seperti *barrier gate*, *box dispenser*, dll. Tentunya, hasil yang didapatkan dari menggunakan *software* yang telah dikembangkan tersebut sudah teruji. Namun permasalahan terjadi ketika seseorang / sebuah perusahaan ingin menggunakan produk tersebut, karena diperlukan *budget* yang cukup besar.

Menjawab permasalahan di atas, pada skripsi kali ini, akan dikembangkan sebuah sistem untuk dapat mengenali dan membaca karakter pada plat nomor kendaraan bermotor umum dengan menggunakan *platform* Raspberry Pi. Dengan menggunakan Raspberry Pi, biaya yang harus dikeluarkan akan menjadi jauh lebih kecil. Di sisi lain, penggunaan Raspberry Pi juga diikuti dengan beberapa batasan, dikarenakan Raspberry Pi merupakan sebuah komputer mini yang memiliki performa tidak sebagus performa komputer pada umumnya. Untuk penelitian ini, metode yang akan digunakan adalah metode YOLO (*You Only Look Once*) dan Tesseract-OCR (*Optical Character Recognition*), dan dilakukan pada platform Raspberry Pi. Metode YOLO dapat melakukan deteksi sebuah objek lebih cepat dan juga akurat [5] dan Tesseract-OCR digunakan untuk membaca / mengambil teks yang terdapat pada gambar.

2. LANDASAN TEORI

2.1 Plat Nomor Kendaraan Bermotor

Tanda nomor kendaraan merupakan salah satu jenis identifikasi kendaraan bermotor. Karakter pada plat nomor di Indonesia terdiri dari 2 hal, yaitu nomor polisi dan juga masa aktif plat nomor tersebut. Untuk nomor polisi, plat nomor Indonesia dimulai dengan huruf yang digunakan untuk membedakan plat nomor tersebut berdasarkan wilayahnya, kemudian diikuti dengan rangkaian angka, dan diakhiri dengan serangkaian huruf [10].

2.2 YOLO

Object detection merupakan salah satu penerapan *machine learning* dalam kehidupan sehari-hari yang paling umum namun menantang untuk dilakukan oleh sebuah sistem [13]. YOLO, singkatan dari *You Only Look Once*, merupakan sebuah metode *object* detection *state of the art* yang diusulkan pada tahun 2015 oleh Redmon et al. Versi YOLO yang akan digunakan pada penelitian ini yaitu YOLOv3. YOLOv3 sendiri merupakan salah satu metode *object detection* yang paling banyak digunakan [20].

YOLO dapat bekerja dengan melihat sebuah gambar satu kali untuk memprediksi objek apa yang terdapat pada gambar tersebut, beserta dengan lokasinya. Hal ini dimungkinkan dengan melihat cara kerja YOLO, yaitu dengan cara memanfaatkan sebuah *convolutional network* untuk memprediksi *bounding box* dan probabilitas *class* pada box-box tersebut secara terus menerus [15].

Untuk cara kerja metode YOLO, sebuah gambar dibagi menjadi *grid* dengan ukuran *S x S*. Jika pada *grid* tersebut terdapat titik pusat suatu objek, maka *grid cell* tersebut bertanggungjawab untuk mendeteksi objek tersebut. Setiap *grid cell* memprediksi *B bounding boxes* dan nilai kepercayaan diri (*confidence scores*) untuk kotak-kotak tersebut. Selain itu, setiap *grid cell* juga memprediksi kemungkinan adanya sebuah objek dari sebuah *class* dalam *grid cell* tersebut.

Setelah mendapatkan nilai-nilai tersebut, model akan dapat memprediksi letak sebuah objek beserta dengan batasannya dan juga *confidence score* model untuk objek tersebut.

2.3 Tesseract-OCR

OCR, singkatan dari *Optical Character Recognition*, merupakan proses konversi dari sebuah gambar yang memiliki teks di dalamnya menjadi sebuah teks yang dapat diedit untuk memprosesnya lebih lanjut [12]. Dengan menggunakan teknologi ini, sebuah mesin dapat langsung mengenali teks secara otomatis. OCR memungkinkan untuk melakukan *edit* teks, mencari sebuah kata / frase, mengimplementasikan teknik seperti penerjemah otomatis, teks ke kemampuan berbicara, dan masih banyak hal lain [2]. Tesseract OCR merupakan teknologi OCR *open source* yang dikembangkan pada tahun 1984 hingga 1994. Namun, baru pada tahun 2005, Tesseract menjadi *open source*. Pada saat ini, pengembangan Tesseract dipegang oleh Google.

2.4 Raspberry Pi

Raspberry Pi merupakan sebuah komputer mini yang diciptakan di Negara Inggris untuk membantu pengajaran *computer science* pada sekolah-sekolah dan negara-negara berkembang. Raspberry Pi telah digunakan untuk banyak hal, seperti robotik, pengembangan super komputer, tablet *portable*, dan masih banyak lagi [9]. Untuk penelitian kali ini, model Raspberry Pi yang akan digunakan adalah Raspberry Pi 3B+ dan Raspberry Pi 4.

Pada penelitian ini juga akan digunakan PiCamera sebagai kamera yang dapat menampilkan *live video stream* dan menangkap gambar. Dengan menggunakan PiCamera yang dipasang pada Raspberry Pi, hal ini dapat menjadi elemen yang krusial pada proyek-proyek yang membutuhkan *real time display* dan *image processing* [4]. Untuk versi PiCamera yang digunakan pada penelitian ini yaitu PiCamera V2 yang memiliki resolusi 8 *megapixel*.

2.5 Metode Pre-Processing

2.5.1 Unsharp Mask

Unsharp mask adalah salah satu teknik *image enhancement* yang paling banyak diketahui dan digunakan [18]. *Unsharp mask* berguna untuk mengurangi tingkat *blur* yang terdapat pada sebuah gambar [1].

2.5.2 Grayscale

Penerapan *grayscale* dalam memproses sebuah gambar dapat mengurangi kebutuhan komputasional [7]. Hal ini tentu akan sangat membantu dalam penelitian ini karena platform yang akan diajdikan tempat untuk menguji adalah sebuah Raspberry Pi, yang tidak memiliki kemampuan komputasional seperti komputer pada umumnya.

2.5.3 Histogram Equalization

Histogram equalization merupakan sebuah teknik untuk mempertajam kontras sebuah gambar. Dengan mempertajam kontras pada gambar yang akan digunakan, maka pengambilan informasi yang ada pada gambar pun akan menjadi lebih mudah dan hasilnya akan menjadi lebih baik jika dibandingkan dengan gambar sebelum diproses [8]. Hal ini dikarenakan ketika sebuah gambar diaplikasikan *histogram equalization*, intensitas dari sebuah *pixel* pada gambar akan didistribusikan secara merata yang akan membuat kualitas gambar menjadi lebih baik [3].

2.5.4 Median Filtering

Pada penelitian ini akan dilakukan *median filtering* pada gambar. Metode ini dilakukan untuk mengurangi *noise* dan *error* yang dapat dimiliki oleh gambar yang akan mengganggu proses pengambilan informasi jika tidak dihilangkan [16]. Hal ini dapat membantu karena terdapat kemungkinan bahwa plat nomor yang akan dibaca kotor karena bekas debu dan sejenisnya.

2.5.5 Gaussian Blur

Gaussian Blur merupakan salah satu metode untuk memperbaiki gambar yang memiliki noise di dalamnya. *Gaussian blur* dapat membuat sebuah gambar menjadi lebih buram dan lebih halus [6]. Pada penelitian ini, *gaussian blur* akan digunakan sebelum mengubah gambar menjadi gambar biner supaya ketika gambar diubah menjadi biner *noise* yang tadinya terdapat pada gambar dapat hilang.

2.5.6 Otsu Binarization

Otsu Binarization merupakan sebuah metode untuk mengkonversi gambar menjadi gambar biner. Sebuah gambar biner merupakan gambar yang hanya memiliki 2 nilai, yaitu 0 dan 1. Untuk mengubah gambar menjadi biner, biasanya akan ditentukan terlebih dahulu sebuah nilai *threshold* yang akan berguna sebagai acuan. Pada *otsu binarization*, nilai *threshold* tersebut akan ditentukan secara otomatis dengan melihat *histogram* pada sebuah gambar [17].

2.5.7 Dilation

Dilation merupakan salah satu jenis *morphological operations*. Pada proses *dilation*, objek-objek akan diperluas. Hal tersebut akan menutup lubang-lubang / objek-objek yang tidak saling sambung [14].

2.6 Pengujian pada Sistem

2.6.1 Intersection over Union

Intersection over Union merupakan sebuah *metric* yang dapat digunakan untuk mengevaluasi performa sebuah *object detection* model. Untuk menghitung IoU, dapat dengan menggunakan rumus sebagai berikut:

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union} \quad (1)$$

Setelah program mendapatkan nilai IoU dari data *ground truth* dan *detected object*, akan dilihat apakah nilai IoU tersebut melebihi *threshold* yang sebelumnya sudah ditentukan. Jika IoU dari perbandingan 2 objek lebih dari *threshold*, maka akan dikategorikan sebagai prediksi positif. Jika IoU dari perbandingan kedua objek kurang dari nilai *threshold*, maka akan dikategorikan sebagai prediksi negatif. Prediksi positif dan negatif akan diklasifikasikan lebih lanjut menjadi *True Positive* (TP), *False Positive* (FP), dan *False Negative* (FN).

2.6.2 Precision

Precision adalah kemampuan sebuah model untuk mengidentifikasi objek yang relevan. Nilai dari *precision* merupakan persentase dari prediksi positif yang benar yang dilakukan oleh model [11]. Jika model memiliki nilai *precision* yang tinggi, maka model dapat membuat banyak klasifikasi positif dengan benar, dan membuat lebih sedikit klasifikasi positif yang salah. Penghitungan *precision* dapat dilakukan dengan rumus:

$$Precision = \frac{True_{positive}}{True_{positive}+False_{positive}} \quad (2)$$

2.6.3 Recall

Recall merupakan kemampuan model untuk menemukan semua objek yang relevan. Nilai dari *recall* menunjukkan presentase dari prediksi positif yang benar dibagi dengan semua *ground truth* yang ada [11]. Semakin tinggi nilai *recall*, menunjukkan bahwa model dapat mendeteksi sampel objek yang positif. Penghitungan *recall* dapat dilakukan dengan rumus:

$$Recall = \frac{True_{positive}}{True_{positive}+False_{negative}} \quad (3)$$

2.6.4 mAP (mean Average Precision)

Mean Average Precision (mAP) merupakan sebuah nilai yang sering digunakan untuk mengukur performa dari sebuah model *object detection*. Nilai dari mAP didapatkan dengan cara mengestimasi luas area yang terdapat di bawah kurva *Precision* dan *Recall* [11].

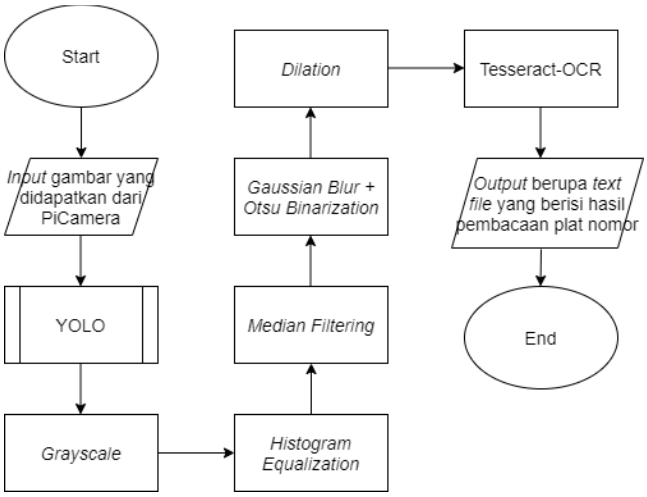
2.6.5 Levenshtein Distance

Levenshtein Distance merupakan salah satu *metric* yang biasa digunakan untuk mengukur jarak dan kemiripan antara 2 buah *string*. Metode ini ditemukan oleh Vladimir Levenshtein pada tahun 1965. Metode ini akan mencari jarak minimal yang harus dilalui oleh sebuah *string* agar dapat menjadi seperti *string* pembandingnya [19].

3. ANALISIS DAN DESAIN SISTEM

3.1 Analisis Sistem

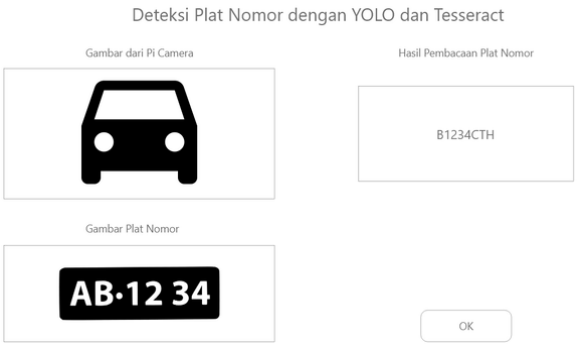
Untuk alur proses program, setelah program mendapatkan *input* gambar berupa gambar hasil jepretan PiCamera, pada gambar tersebut akan dilakukan metode YOLO untuk mendeteksi plat nomor pada mobil yang difoto. Setelah itu gambar plat nomor akan melalui beberapa tahapan *pre-processing*. Jika gambar sudah melewati proses tersebut, gambar akan dijadikan sebagai *input* untuk Tesseract-OCR untuk dibaca. Tesseract-OCR akan mengeluarkan *output* berupa *text file* yang berisi hasil pembacaan plat nomor. Untuk *flowchart* alur proses program dapat dilihat pada Gambar 1.



Gambar 1. *Flowchart* system

3.2 Desain Sistem

Untuk *interface* dari sistem akan dibuat dengan menggunakan tkinter. Pada *interface* sistem, akan terdapat 3 tampilan yang akan ditampilkan. Frame pertama adalah *video stream* dari Pi Camera yang akan menampilkan secara langsung apa yang ada di depan Pi Camera. Kemudian terdapat gambar hasil jepretan dari *frame* pertama yang akan menunjukkan foto mobil yang terlihat plat nomornya. Lalu gambar terakhir yaitu gambar plat nomor yang telah dideteksi oleh YOLO beserta dengan pembacaan plat nomornya. Rencana tampilan UI program dapat dilihat pada Gambar 2.



Gambar 2. Rencana tampilan UI

4. PENGUJIAN SISTEM

4.1 Evaluasi Model YOLO

Pada pengujian pertama, akan dilakukan evaluasi untuk model YOLO yang digunakan. Pengujian model akan dilakukan dengan menghitung nilai mAP (*mean Average Precision*) untuk *class* 'plat nomor'. Pengujian dilakukan pada 20 gambar yang belum pernah dilihat oleh model saat proses *training*. Model akan diuji pada 3 nilai *threshold* IoU, yaitu 50%, 60%, dan 70%.

Pada nilai *threshold* 50%, nilai mAP mencapai **100%**. Selain itu, dalam pengujian pada nilai ini, dari 20 gambar yang diuji, model dapat mendeteksi 20 objek dan semua objek yang terdeteksi dikategorikan sebagai *true positive*.

Pada nilai *threshold* 60%, nilai mAP mencapai **94,5%**. Selain itu, dalam pengujian pada nilai ini, dari 20 gambar yang diuji, model dapat mendeteksi 20 objek, yang terbagi menjadi 19 objek dikategorikan sebagai *true positive* dan hanya 1 objek yang dikategorikan menjadi *false positive*.

Pada nilai *threshold* 70%, nilai mAP mencapai **80,31%**. Dalam pengujian pada nilai *threshold* 70% ini, dari 20 gambar yang diuji, model dapat mendeteksi 20 objek, yang terbagi menjadi 17 objek yang dikategorikan sebagai *true positive* dan 3 objek yang dikategorikan menjadi *false positive*.

4.2 Implementasi Metode pada Gambar

4.2.1 Unsharp Mask

Pada penelitian ini, gambar yang diambil melalui PiCamera akan diaplikasikan *unsharp mask* untuk mempertajam kualitas gambar agar dapat diolah dengan lebih baik kedepannya oleh program.

Gambar 3. Contoh gambar yang diambil dengan PiCamera

Gambar 4. Gambar yang telah diaplikasikan *unsharp mask*

4.2.2 YOLO

Pada program, gambar yang telah dijadikan sebagai *input* pertama-tama akan dilakukan metode YOLO untuk mendeteksi apakah ada plat nomor pada gambar tersebut atau tidak.

Gambar 5. Contoh hasil pemotongan gambar

4.2.3 Grayscale

Setelah program telah mendapatkan gambar plat nomor pada gambar awal, akan diaplikasikan *grayscale* agar dapat meringankan proses komputasional yang harus dilakukan oleh Raspberry Pi pada tahapan selanjutnya.

Gambar 6. Hasil konversi gambar ke *grayscale*

4.2.4 Histogram Equalization

Setelah gambar plat nomor dikonversi menjadi *grayscale*, program akan mengaplikasikan *histogram equalization* untuk meningkatkan kontras pada gambar.

Gambar 7. Hasil penerapan *histogram equalization*

4.2.5 Median Filtering

Untuk tahapan selanjutnya, program akan mengaplikasikan *median filtering*. Penerapan metode ini bertujuan untuk mengurangi noise yang terdapat pada gambar.

Gambar 8. Hasil penerapan *median filtering*

4.2.6 Gaussian Blur + Otsu Binarization

Setelah pada gambar diaplikasikan *median filtering*, selanjutnya gambar akan diubah menjadi biner agar dapat dibaca oleh Tesseract OCR. Namun, sebelum mengubah gambar menjadi biner, pada gambar akan dilakukan *Gaussian Blur* terlebih dahulu untuk mengurangi *noise* pada gambar sebelum diubah menjadi biner.

Gambar 9. Hasil konversi gambar menjadi biner

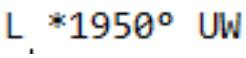
4.2.7 Dilation

Metode terakhir yang akan diaplikasikan kepada gambar yaitu *dilation*, yang berguna untuk memperbesar area objek pada gambar untuk memperjelasnya, pada gambar ini objek yang dimaksud merupakan karakter yang terdapat pada plat nomor.

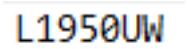
Gambar 10. Hasil penerapan *dilation* pada gambar

4.2.8 Tesseract-OCR

Gambar 11 menunjukkan hasil pembacaan plat nomor yang dilakukan oleh program. Namun, pada pembacaan pertama oleh program, terkadang ditemukan beberapa karakter yang seharusnya tidak diikutkan, seperti karakter '*' dan 'º' pada Gambar 11. Untuk mengatasi hal tersebut, program akan melakukan pembersihan karakter-karakter tersebut dengan cara hanya mengikutkan karakter yang berupa huruf kapital dan angka. Hasil dari pembersihan ini dapat dilihat pada Gambar 12.



Gambar 11. Hasil pembacaan pertama program



Gambar 12. Hasil pembersihan karakter oleh program

4.3 Perbandingan Hasil Pengujian

Pada pengujian kali ini, pengujian akan dilakukan pada 20 gambar mobil dan akan dihitung tingkat akurasi untuk setiap karakter yang terdapat pada plat nomor. Kemudian, akan dilakukan rata-rata terhadap akurasi pembacaan karakter plat nomor 20 gambar tersebut.

4.3.1 Kombinasi Metode Pre-processing

4.3.1.1 Grayscale

Untuk pengujian pertama, pada gambar akan dikonversi menjadi *grayscale* kemudian gambar akan diubah menjadi biner untuk dibaca oleh Tesseract. Untuk pembacaan karakter plat nomor dengan metode *pre-processing* hanya *grayscale*, akurasi rata-rata yang didapatkan mencapai **71,46%**.

4.3.1.2 Grayscale + Histogram Equalization

Untuk percobaan kedua, metode *pre-processing* yang diaplikasikan ke gambar yaitu *grayscale*, yang diikuti dengan *Histogram Equalization*, yang kemudian diubah menjadi biner. Akurasi pembacaan plat nomor yang didapatkan dengan menerapkan metode *pre-process* ini yaitu mencapai **64,97%**.

4.3.1.3 Grayscale + Histogram Equalization + Median Filtering

Untuk percobaan ketiga, akan dilakukan 3 metode *pre-processing*, yaitu dimulai dengan menkonversi gambar menjadi *grayscale*, dilanjutkan dengan mengaplikasikan *histogram equalization*, dan terakhir diaplikasikan *median filtering*, sebelum diubah menjadi gambar biner untuk dibaca oleh Tesseract. Hasil akurasi pembacaan plat nomor yang didapatkan dengan menerapkan ketiga metode tersebut hanya mencapai **53,45%**.

4.3.1.4 Tanpa Menggunakan Dilation

Pada percobaan terakhir, akan dilakukan metode *pre-processing grayscale*, namun ketika akan dibaca oleh program, gambar hanya akan diubah menjadi biner, tanpa diaplikasikan *dilation* terlebih dahulu. Hasil akurasi pembacaan plat nomor yang didapat mencapai nilai **67,08%**.

4.3.2 Kamera yang Digunakan

Pada perbandingan kedua, akan dilakukan perbandingan hasil pengujian ketika program menggunakan gambar yang diambil dengan menggunakan kamera *handphone* dengan gambar yang diambil dengan menggunakan PiCamera. Untuk hasil pembacaan program dengan menggunakan gambar yang diambil dengan menggunakan kamera *handphone* mencapai nilai **71,46%**. Untuk hasil pembacaan program dengan menggunakan gambar yang diambil dengan menggunakan PiCamera, rata-rata akurasi pembacaan plat nomor dengan menggunakan PiCamera mencapai nilai **62,89%**.

4.3.3 Pencahayaan pada Gambar

Pada perbandingan kali ini, akan dilakukan pengujian dengan situasi pencahayaan yang berbeda. Terdapat foto yang diambil pada waktu siang hari dan terdapat foto yang diambil pada malam hari, dengan *flash camera* maupun tidak menggunakan *flash*. Untuk foto yang diambil pada siang hari ketika situasi pencahayaan masih bagus, tingkat akurasi pembacaan plat nomor mencapai **77,18%**. Untuk hasil pembacaan karakter foto yang diambil pada malam hari dengan menggunakan *flash*, tingkat akurasi mencapai **54,29%**. Namun, untuk hasil pembacaan karakter foto yang diambil pada malam hari tanpa menggunakan *flash*, tingkat akurasi hanya mencapai **42.86%**.

4.3.4 Perbedaan Versi Raspberry Pi

Waktu yang dibutuhkan untuk menjalankan program pada Raspberry Pi 4 rata-rata **11.55** detik, sementara waktu yang dibutuhkan untuk menjalankan program pada Raspberry Pi 3B+ rata-rata **61.72** detik. Sementara itu untuk akurasi pembacaan karakter pada kedua versi Raspberry Pi sama.

4.4 Hasil Implementasi *Interface*

Tampilan *interface* program dapat dibagi menjadi 3 bagian utama. Program dapat menampilkan *video stream* dari PiCamera, kemudian program dapat mengambil tangkapan layar dari *video stream* tersebut dan akhirnya gambar dapat diproses oleh program sehingga dapat menunjukkan hasil pembacaan plat nomor. Untuk gambar tampilan program dapat dilihat pada Gambar 13, Gambar 14, dan Gambar 15.

Gambar 13. Tampilan awal *interface* program menampilkan *video stream* PiCamera

Gambar 14. Tampilan ketika program menangkap gambar dari PiCamera

Gambar 15. Tampilan ketika program telah berhasil membaca plat nomor

5. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil perancangan dan pembuatan program, dapat diambil kesimpulan bahwa:

- Penggunaan Raspberry Pi sebagai *platform* untuk mengembangkan sebuah teknologi yang lebih murah dapat dilakukan, dengan bantuan metode YOLO untuk mendeteksi objek dan Tesseract-OCR untuk membaca plat nomor.
- Model YOLO yang sudah dilatih memiliki tingkat presisi yang tinggi dalam pengenalan objek berupa plat nomor, yang mencapai nilai **100%** pada nilai threshold 50%; **94,5%** pada nilai threshold 60%; dan **80,31%** pada nilai threshold 70%. Untuk waktu yang dibutuhkan program dalam menjalankan YOLO, dibutuhkan waktu rata-rata **7,11** detik.
- Pre-processing pada gambar mempengaruhi hasil dari pembacaan karakter pada plat nomor, namun tidak semua metode pre-processing akan meningkatkan hasil pembacaan karakter. Jika pada gambar diaplikasikan histogram equalization, maka tingkat akurasi akan menurun **6,49%**. Kemudian jika pada gambar diaplikasikan histogram equalization dan median filtering, maka tingkat akurasi akan menurun **18,01%**. Namun, ketika pada gambar dilakukan metode dilation, tingkat akurasi akan meningkat sebesar **4,37%**.
- Program yang telah dibuat dapat dijalankan pada Raspberry Pi 4 dengan memiliki tingkat akurasi pembacaan karakter mencapai **71,46%** dan rata-rata membutuhkan waktu selama **11,55** detik. Ketika program dijalankan pada Raspberry Pi 3B+, tingkat akurasi yang didapatkan sama, namun program dijalankan dengan waktu yang lebih lama, yaitu **61,72** detik. Ketika menggunakan Raspberry Pi 3B+, terkadang program akan tidak mengeluarkan hasil sama sekali. Hal ini dikarenakan Raspberry Pi 3B+ tidak memiliki spesifikasi hardware seperti Raspberry Pi 4.
- Kondisi pencahayaan pada gambar mempengaruhi hasil dari pembacaan plat nomor. Pada gambar yang diambil pada siang hari (pencahayaan bagus), program dapat mendapatkan rata-rata akurasi **77,18%.** Namun, pada malam hari, jika gambar diambil dengan menggunakan flash (pencahayaan kurang), program masih mendapatkan rata-rata akurasi senilai **54,29%.** Untuk pengambilan gambar pada malam hari tanpa menggunakan flash (pencahayaan sangat kurang), program hanya akan mendapat nilai rata-rata akurasi sebesar **42,86%.**
- Kamera yang digunakan untuk mengambil gambar juga akan mempengaruhi hasil program. Jika kamera yang digunakan lebih bagus, maka hasil pembacaan oleh program akan otomatis lebih baik. Pada penelitian ini, ketika gambar diambil menggunakan kamera handphone yang memiliki kualitas lebih bagus daripada PiCamera dan semua variabel lainnya sama, rata-rata akurasi yang didapatkan lebih tinggi **8,57%** dibandingkan jika gambar diambil dengan menggunakan PiCamera.
- Untuk noise yang terkadang timbul pada gambar plat nomor kendaraan bermotor umum yaitu karakter plat nomor yang kurang tebal, dapat diatasi dengan menggunakan dilation.

5.2 Saran

Saran yang dapat diberikan untuk menyempurnakan dan mengembangkan skripsi ini lebih lanjut antara lain:

- Untuk meningkatkan kecepatan program dapat menggunakan versi YOLO yang lebih ringan dari YOLOv3, yaitu Tiny-YOLO. Namun, dengan menggunakan Tiny-YOLO, akan ada *trade-off* yaitu tingkat akurasi deteksi objek yang lebih rendah daripada YOLOv3.
- Mencari dan mengimplementasikan metode *pre-processing* yang tepat agar akurasi pembacaan karakter pada plat nomor menjadi lebih tinggi.

6. DAFTAR PUSTAKA

[1] Al-Ameen, Z., Muttar, A., & Al-Badrani, G. 2019. Improving the Sharpness of Digital Image Using an Amended Unsharp Mask Filter. *International Journal of Image, Graphics and Signal Processing*, *11*(3), 1–9. https://doi.org/10.5815/ijigsp.2019.03.01

[2] Ali, N., Isheawy, M., & Hasan, H. 2015. Optical Character Recognition (OCR) System. *IOSR Journal of Computer Engineering Ver. II*, *17*(2), 2278–2661. https://doi.org/10.9790/0661-17222226

[3] Azam, M. A. 2016. A Review of Various Histogram Equalization Techniques for Image Enhancement. *IOSR Journal of Electrical and Electronics Engineering*, *11*(04), 48–51. https://doi.org/10.9790/1676-1104044851

[4] Bowman, R. W., Vodenicharski, B., Collins, J. T., & Stirling, J. 2020. Flat-Field and Colour Correction for the Raspberry Pi Camera Module. *Journal of Open Hardware*, *4*(1), 1–9. https://doi.org/10.5334/joh.20

[5] Du, J. 2018. Understanding of Object Detection Based on CNN Family and YOLO. *Journal of Physics: Conference Series*, 1004(1). https://doi.org/10.1088/1742-6596/1004/1/012029

[6] Gedraite, E. S., & Hadad, M. 2011. Investigation on the effect of a Gaussian Blur in image filtering and segmentation. *Proceedings Elmar - International Symposium Electronics in Marine*, *January 2011*, 393–396.

[7] Kanan, C., & Cottrell, G. W. 2012. Color-to-grayscale: Does the method matter in image recognition? PLoS ONE, 7(1). https://doi.org/10.1371/journal.pone.0029740

[8] Mau, S. D. B. 2016. Pengaruh Histogram Equalization Untuk Perbaikan Kualitas Citra Digital. Simetris : *Jurnal Teknik*

Mesin, Elektro Dan Ilmu Komputer, *7*(1), 177. https://doi.org/10.24176/simet.v7i1.502

[9] Nayyar, A., & Puri, V. 2015. Raspberry Pi-A Small , Powerful , Cost Effective and Efficient Form Factor Computer : A Review International Journal of Advanced Research in Raspberry Pi- A Small , Powerful , Cost Effective and Efficient Form Factor Computer : A Review. December. https://www.researchgate.net/profile/Anand_Nayyar/publication/305668622_Raspberry_Pi-A_Small_Powerful_Cost_Effective_and_Efficient_Form_Factor_Computer_A_Review/links/5798c41908aeb0ffcd08b80f/Raspberry-Pi-A-Small-Powerful-Cost-Effective-and-Efficient-Form

[10] Nurhaida, I., Nududdin, I., & Ramayanti, D. 2020. Indonesian license plate recognition with improved horizontal-vertical edge projection. *Indonesian Journal of Electrical Engineering and Computer Science*, *21*(2), 811–821. https://doi.org/10.11591/ijeecs.v21.i2.pp811-821

[11] Padilla, R., Netto, S. L., & Da Silva, E. A. B. 2020. A Survey on Performance Metrics for Object-Detection Algorithms. *International Conference on Systems, Signals, and Image Processing*, *2020-July*(July), 237–242. https://doi.org/10.1109/IWSSIP48289.2020.9145130

[12] Patel, C., Patel, A., & Patel, D. 2012. Optical Character Recognition by Open source OCR Tool Tesseract: A Case Study. *International Journal of Computer Applications*, *55*(10), 50–56. https://doi.org/10.5120/8794-2784

[13] Porikli, F., & Yilmaz, A. 2012. Object detection and tracking. In Studies in Computational Intelligence (Vol. 409, Issue January). https://doi.org/10.1007/978-3-642-28598-1_1

[14] Raid, A. ., Khedr, W. ., El-dosuky, M. ., & Aoud, M. 2014. Image Restoration Based on Morphological Operations. *International Journal of Computer Science, Engineering and Information Technology*, *4*(3), 9–21. https://doi.org/10.5121/ijcseit.2014.4302

[15] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. 2016. You only look once: Unified, real-time object detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016-Decem*, 779–788. https://doi.org/10.1109/CVPR.2016.91

[16] Villar, S. A., Torcida, S., & Acosta, G. G. 2017. Median Filtering: A New Insight. *Journal of Mathematical Imaging and Vision*, *58*(1), 130–146. https://doi.org/10.1007/s10851-016-0694-0

[17] Yousefi, J. 2015. Image Binarization using Otsu Thresholding Algorithm. *Research Gate*, *May*. https://doi.org/10.13140/RG.2.1.4758.9284

[18] Zaafouri, A., Sayadi, M., & Fnaiech, F. 2011. A developed unsharp masking method for images contrast enhancement. *International Multi-Conference on Systems, Signals and Devices, SSD'11 - Summary Proceedings*, *November 2017*. https://doi.org/10.1109/SSD.2011.5767378

[19] Zhang, S., Hu, Y., & Bian, G. 2017. Research on string similarity algorithm based on Levenshtein Distance. *Proceedings of 2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference, IAEAC 2017*, *1*, 2247–2251. https://doi.org/10.1109/IAEAC.2017.8054419

[20] Zhao, L., & Li, S. 2020. Object detection algorithm based on improved YOLOv3. *Electronics (Switzerland)*, *9*(3). https://doi.org/10.3390/electronics9030537