KALKULATOR SAINTIFIK BERBASIS KAMERA

Nenden Siti Fathonah[*], Achmad Yogie Pratama[**], Fajar Astuti Hermawati[**]
[*]Teknik Informatika, Universitas Mercubuana Jakarta, Indonesia
[**]Teknik Informatika, Universitas 17 Agustus 1945 Surabaya, Indonesia
E-Mail: [*]nenden@mercubuana.ac.id, [**]fajarastuti@untag-sby.ac.id

ABSTRAK

Sistem *Optical Character Recognition* (OCR) merupakan teknologi pengolahan citra untuk mengidentifikasi tulisan atau karakter yang terdapat pada sebuah gambar. Dalam menerjemahkan suatu citra, *Optical Character Recognition* melakukan segmentasi terlebih dahulu terhadap citra tersebut sehingga menjadi potongan – potongan gambar karakter. Setelah terbagi menjadi potongan – potongan gambar sistem OCR melakukan pengenalan pada masing – masing gambar karakter tersebut. Karakter dalam gambar yang di scan dengan OCR diubah menjadi text yang kemudian ditampilkan ke layar. Penelitian ini mengimplementasikan *template matching* dengan koefisien *correlation* untuk mengidentifikasi tulisan atau text yang terdapat pada sebuah citra. Dan kemudian hasil keluaran dari OCR akan diubah menjadi angka dan operator untuk selanjutnya dilakukan proses kalkulasi atau penghitungan. Dan hasil perhitungan kemudian ditampilkan ke layar. Dari beberapa percobaan diperoleh akurasi pengenalan dan perhitungan sebesar 85%.

Kata Kunci: OCR, *template matching*, *correlation*

1. Pendahuluan

Keinginan manusia yang ingin serba cepat termasuk dalam hal menghitung membuat banyak software atau alat bantu hitung diciptakan. Kalkulator elektronik merupakan perangkat elektronik portabel yang digunakan untuk melakukan perhitungan, mulai dari aritmatika dasar hingga matematika kompleks.

Sekarang ini, hampir setiap orang menggunakan kalkulator dalam kehidupannya untuk melakukan perhitungan. Namun dari semua software atau alat bantu hitung yang diciptakan semuanya masih harus diinputkan secara manual oleh pengguna. Hal ini terkadang membuat orang cenderung malas untuk menggunakannya. Untuk menjawab semua itu kalkulator berbasis kamera dapat menjadi solusi untuk mengatasinya. Pengguna tidak perlu repot-repot untuk menginputkan angka ke dalamnya.

Aplikasi kalkulator berbasis kamera ini termasuk ke dalam jenis Optical Character Recognition (OCR) yang mengidentifikasi gambar tulisan soal perhitungan matematika dan mengubahkan ke dalam bentuk karakter-karakter yang dapat dikomputasi guna mendapatkan hasil perhitungannya. Optical Character Recognition (OCR) adalah proses yang memungkinkan suatu sistem tanpa campur tangan manusia mengidentifikasi skrip atau huruf yang ditulis ke dalam komunikasi verbal pengguna [1]–[4]. Teknologi ini digunakan di berbagai aplikasi yang menerapkan scanning sebagai fiturnya. Dengan memanfaatkan teknologi OCR ini dapat dilakukan scanning pada

gambar soal perhitungan lalu kemudian diubah menjadi karakter dan langsung otomatis melakukan perhitungan sehingga pengguna tidak perlu lagi memasukkan angka dan juga operator untuk menghitung.

Pada kalkulator berbasis kamera ini, digunakan metode klasifikasi dengan memanfaatkan metode *Template* Matching yang merupakan proses membandingkan gambar dengan menggunakan stored *template*, salah satunya adalah *Template* Matcing Correlation. Pencocokan *template* merupakan metode dimana dengan diberikan gambar referensi dari objek target, mencari apakah objek target itu ada dalam gambar adegan di bawah pemrosesan gambar, dan menemukan lokasinya. Pencocokan korelasi silang memberikan posisi lokal objek dalam gambar keseluruhan[5].

2. Tinjauan Pustaka

2.1. Optical Character Recognition

Optical Character Recognition (OCR) merupakan sebuah sistem yang menterjemahkan secara elektronik atau mekanis dari gambar tulisan tangan atau teks cetak ke dalam teks yang dapat diedit oleh mesin [1]. Banyaknya aplikasi yang membutuhkan teknologi OCR menyebabkan penelitian di bidang ini merupakan salah satu penelitian yang terus berkembang. Dengan OCR yang dikombinasi dengan teknologi terkini seperti smartphone yang dilengkapi dengan kamera, ada banyak pekerjaan yang dapat diselesaikan dengan mudah dan cepat. Seperti contohnya dapat digunakan untuk memindai angka pada meter PLN dan PDAM, memindai tulisan pada kemasan makanan, memindai dokumen kerja dan masih banyak lagi.

Beberapa penelitian tentang OCR menggunakan metode-metode dalam pengenalan pola atau mesin pembelajaran. Hermawati & Koesdijarto [6] menggunakan metode Hidden Markov Model untuk mengenali karakter pada plat nomor kendaraan Indonesia. Singla & Yadav [1] menerapkan metode *template* matching untuk mengenali karakter pada sebuah teks dan merubahnya menjadi suara. Sementara itu, Apriyanti & Widodo [3] menggunakan metode jaringan syaraf tiruan untuk mengenali karakter pada teks dan dirubah ke suara pada perangkat android. Jaringan syaraf tiruan juga digunakan oleh Dongare dkk [7] untuk mengenali karakter Devanagari dalam bentuk tulisan tangan serta oleh Anugrah dan Bintoro [8] untuk mengenali karakter pada media cetak dan merubahnya menjadi bentuk digital. Dengan menggabungkan metode jaringan syaraf tiruan dan *binary pattern*, Raghavendra & Danti [9] mengenali karakter pada cek bank India.

2.2. *Template* Matching

Pencocokan *template* (*template matching*) adalah sebuah teknik dimana jika diberi gambar referensi dari objek target, sistem akan mencari apakah objek target itu ada dalam gambar keseluruhan di bawah pemrosesan gambar, dan menemukan lokasinya [5]. Masalah pencocokan terdiri dalam menentukan parameter transformasi yang tidak diketahui untuk memetakan gambar *template* untuk mencocokkan gambar sumber. Parameter transformasi yang tidak diketahui adalah terjemahan, rotasi, skala, dan kemiringan gambar. Penelitian pencocokan pola tradisional memiliki banyak aplikasi seperti korelasi titik, pencocokan talang

(chamfer matching), dan pencocokan adegan hirarki berurutan. Seperti yang dilakukan Hermawati dan Sholeh [10] yang menerapkan *template* matching untuk mengenali rambu batas kecepatan pada sebuah citra.

Algoritma *template matching* dengan koefisien korelasi tersebut akan dijabarkan sebagai berikut [11] :

$$r = \frac{\sum_{k=1}^{n}(x_{ik}-\bar{x}_i).(x_{jk}-\bar{x}_j)}{\sqrt{[\sum_{k=1}^{n}(x_{ik}-\bar{x}_i)^2.\sum_{k=1}^{n}(x_{jk}-\bar{x}_j)^2.]}} \quad (1)$$

dimana, $\bar{x}_i$ dan $\bar{x}_j$ merupakan rata-rata dari matriks *i* dan *j* yang dapat dihitung dengan:

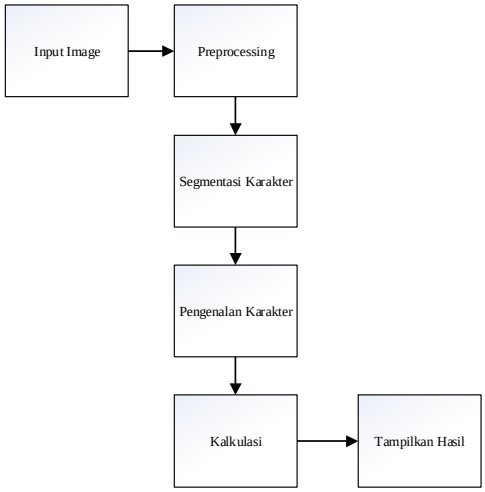
$$\bar{x}_i = \frac{1}{n}\sum_{k=1}^{n} x_{ik} \quad (2)$$

$$\bar{x}_j = \frac{1}{n}\sum_{k=1}^{n} x_{jk} \quad (3)$$

dengan *r* = nilai korelasi antara matriks *i* dan *j*, x_{ik} = nilai pixel ke-*k* pada matriks *i*, x_{jk} = nilai pixel ke-*k* pada matriks *j*, dan *n* = jumlah pixel pada suatu matriks.

3. Metode

Metode yang diterapkan dalam penelitian ini dibagi menjadi beberapa tahap seperti yang disajikan pada Gambar 1.

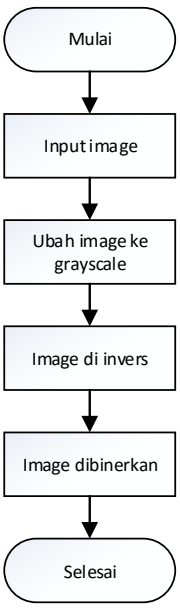


Gambar 1. Blok diagram sistem

Langkah pertama merupakan tahap pra proses untuk merubah citra input ke dalam bentuk citra grayscale. Selanjutnya dilakukan proses segmentasi untuk mendapatkan potongan gambar masing-masing karakter. Setiap karakter yang diperoleh, dikenali dengan menggunakan algoritma *template* matching. Teks yang diperoleh kemudian dikalkulasi untuk mendapatkan hasil perhitungan dari rumus yang diberikan dalam citra input.

3.1. Preprosessing

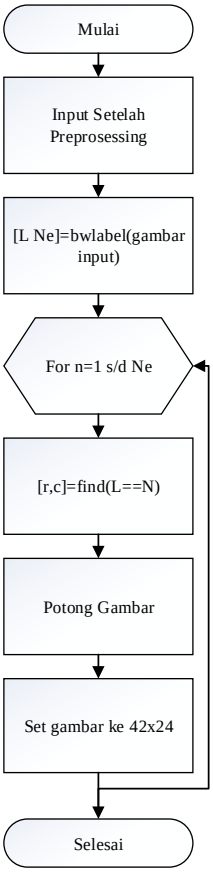
Tahap *preprocessing* seperti pada Gambar 2, merupakan tahapan awal untuk mengubah warna citra dari RGB ke model grayscale. Kemudian dilakukan tresholding untuk mendapatkan citra biner yaitu citra yang hanya memiliki dua warna yaitu hitam dan putih. Hasil dari tresholding kemudian dilakukan invers untuk memperoleh citra dengan background hitam dan foreground putih.



Gambar 2. *Flowchart* tahap *preprocessing*

3.2. Segmentasi Karakter

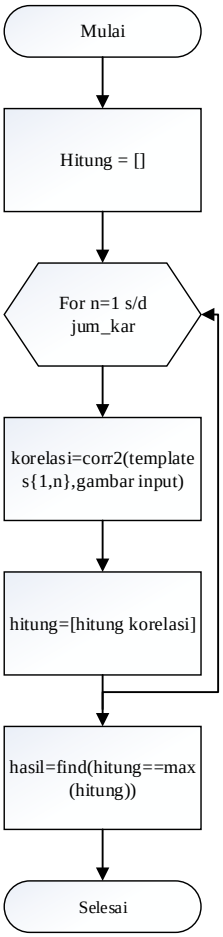
Setelah dilakukan *preprosessing* tahap berikutnya adalah segmentasi karakter. Segementasi karakter ini dilakukan untuk memperoleh potongan potongan gambar karakter yang akan digunakan untuk pengenalan karakter. Pada citra input hasil preprosessing dihitung komponen – komponen yang terhubung pada foreground menggunakan fungsi bwlabel. Lalu dilakukan perulangan dari *n* = 1 sampai jumlah komponen tersebut dan kemudian potong gambar. Setelah gambar dipotong kemudian dilakukan resize menjadi 42x24 agar sama seperti gambar yang ada di *template*. Langkah-langkah dalam segmentasi karakter dapat dilihat pada Gambar 3.



Gambar 3. *Flowchart* tahap segmentasi karakter

3.3. Pengenalan Karakter

Setelah dilakukan segmentasi berikutnya adalah tahap pengenalan karakter. Dalam tahapan ini terdapat dua langkah yaitu pembuatan matriks *template* dan pencocokan input dengan *template*. Gambar 4 menunjukkan tahapan dalam langkah pencocokan.



Gambar 4. *Flowchart* tahap pencocokan karakter

Langkah pertama dalam tahapan ini adalah membuat matriks *template* yang memuat gambar karakter yang akan digunakan untuk pencocokan. Pertama disiapkan gambar – gambar karakter yang akan dipakai untuk pencocokan. Selanjutnya buat matriks letter yang berisi huruf alphabet dari huruf A , C , E , G , I , L , N , O , P , S , T , X. Huruf yang dipakai disini adalah huruf capital karena tujuannya

adalah untuk membedakan beberapa karakter huruf yang memiliki kemiripan dengan karakter angka. Seperti contohnya huruf l dan angka 1. Karena itulah dipakai huruf capital karena huruf L capital berbeda dengan angka 1. Berikutnya dibuat matriks number yang berisi angka 0 sampai angka 9 kemudian dilanjutkan dengan operator jumlah (+), kurang (-), kali (*), bagi (/), pangkat (^), dan akar (√). Setelah itu kedua matriks tersebut digabung menjadi satu kesatuan menjadi matriks character.

Dan setelah membuat *template* selanjutnya dilakukan tahap pencocokan untuk mengetahui apakah citra input memiliki kecocokan dengan gambar yang ada pada matriks *template*, seperti flowchart pada Gambar 4.

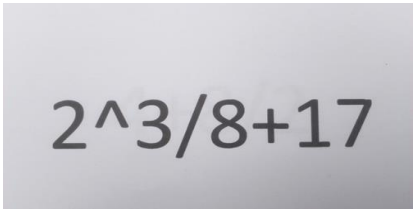
3.4. Kalkulasi

Setelah dilakukan pengenalan karakter hasil output dari tahap pengenalan karakter kemudian dilakukan proses perhitungan. Hasil dari proses pengenalan karakter ini adalah berupa string. Karena itu digunakan inline function untuk menyimpannya. Kemudian dari inline function dipanggil menggunakan fungsi ans untuk mengkalkulasi hasil perhitungan tersebut. Dan terakhir, hasil dari perhitungan kemudian akan ditampilkan ke layar.

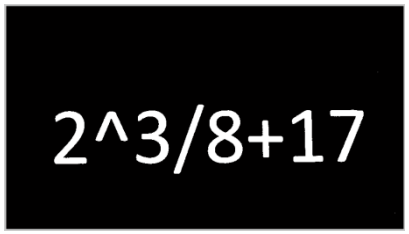
4. Hasil dan Pembahasan

4.1 Uji Coba *Preprosessing*

Pengujian pertama dilakukan pengujian *preprocessing* untuk mengetahui seberapa besar pengaruh cahaya pada proses ini. Percobaan pertama dengan pencahayaan yang baik seperti pada Gambar 5, dimana hasil preprocessingnya ditunjukkan pada Gambar 6.



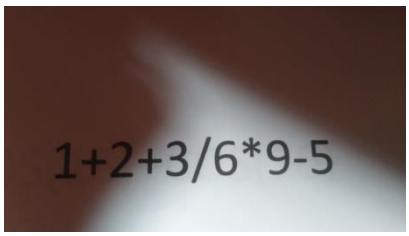
Gambar 5. Citra Input (RGB)



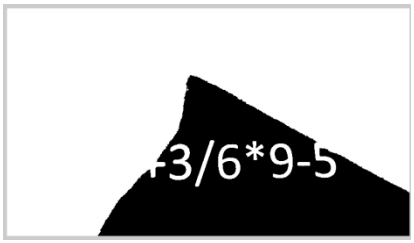
Gambar 6. Hasil Preprosessing

Dari hasil pengujian ini diketahui bahwa pencahayaan yang baik akan menghasilkan preprosessing yang baik pula.

Pengujian berikutnya dilakukan pada gambar dengan kondisi pencahayaan yang tidak merata, seperti pada Gambar 7. Hasil dari tahap preprocessing seperti terlihat pada Gambar 8, terdapat bagian tulisan yang tidak terlihat karena berada pada daerah dengan pencahayaan rendah.



Gambar 7. Citra Input (RGB)



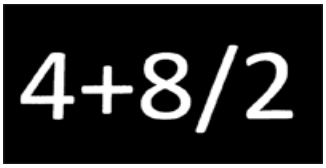
Gambar 8. Hasil Preprosessing

Dari hasil pengujian ini terlihat bahwa preprosessing tidak akan

berhasil dengan baik jika gambar yang diambil memiliki pencahayaan yang tidak merata.

4.3. Uji Coba Segmentasi

Uji coba ini dilakukan untuk mengetahui seberapa besar pengaruh sudut pada proses segmentasi. Percobaan pertama adalah dengan pengambilan gambar posisi tegak lurus, seperti pada Gambar 9. Citra dengan posisi tegak lurus dapat menghasilkan potongan gambar atau hasil segmentasi yang baik seperti pada Gambar 10.



Gambar 9. Citra Input (Biner)

Gambar 10. Hasil Segmentasi

Dari hasil pengujian ini dapat diketahui bahwa pengambilan gambar dengan sudut yang baik dapat menghasilkan segmentasi yang baik pula.

Pengujian selanjutnya dilakukan dengan mengambil gambar tidak dengan posisi tegak lurus namun terdapat kemiringan sebesar 30-45 derajat, seperti pada Gambar 11 dengan hasil segmentasi pada Gambar 12.

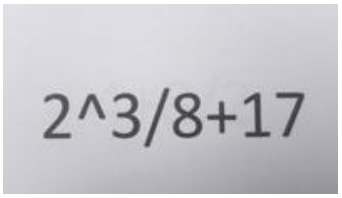
Gambar 11. Citra Input (Biner)

Gambar 12. Hasil Segmentasi

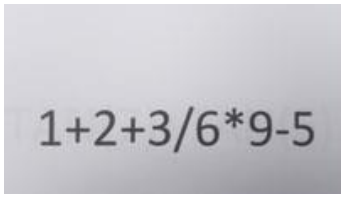
Dari Gambar 12 dapat dilihat bahwa gambar yang diambil secara miring dapat menyebabkan kegagalan pada proses segmentasi yang berakibat pada tidak berhasilnya proses pengenalan karakter.

4.4. Uji Coba Pengenalan Karakter

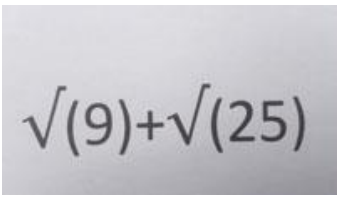
Pada uji coba pengenalan karakter dilakukan pengujian pada soal – soal perhitungan dengan kondisi pencahayaan yang baik. Data uji coba yang akan digunakan adalah gambar soal perhitungan berupa citra RGB. Uji coba ini dilakukan pada 20 gambar soal. Uji coba ini dilakukan untuk mengetahui seberapa akurat sistem dalam mengenali karakter karakter dari gambar soal yang diujikan. Proses ujicoba pada gambar dapat dilihat pada Gambar 13.



Hasil = 2^3/8+17



Hasil = 1+2+3/6*9-5



Hasil=sqrt(9)+sqrt(25)

Gambar 13. Uji Coba Pengenalan Karakter

Hasil ujicoba pengenalan karakter secara lengkap disajikan pada Tabel 1.

Tabel 1. Hasil Pengujian Pengenalan Karakter

Gambar	Data	Hasil	Benar
Soal 1	2^3/8+17	2^3/8+17	Ya
Soal 2	SIN(20)+SIN(15)	sin(2o)+sin(15)	Tidak
Soal 3	1+2+3/6*9-5	1+2+3/6*9-5	Ya
Soal 4	TAN(7)+TAN(5)	tan(7)+tan(5)	Ya
Soal 5	COS(15)+COS(10)	cos(15)+cos(1o)	Tidak
Soal 6	√(9)+√(25)	sqrt(9)+sqrt(25)	Ya
Soal 7	LOG(2)+LOG(3)	log(2)+log(3)	Ya
Soal 8	COS(25)+COS(10)	cos(25)+cos(1o)	Tidak
Soal 9	44+33-77	44+33-77	Ya
Soal 10	7*4-3+5	7*4-3+5	Ya
Soal 11	SIN(3)+SIN(4)	sin(3)+sin(4)	Ya
Soal 12	EXP(2)+EXP(3)	exp(2)+exp(3)	Ya
Soal 13	4+8/2	4+8/2	Ya
Soal 14	LOG(20)	log(20)	Ya
Soal 15	COS(60)	cos(60)	Ya
Soal 16	EXP(20)	exp(20)	Ya
Soal 17	TAN(10)	tan(10)	Ya
Soal 18	LOG(3)+LOG(5)	log(3)+log(5)	Ya
Soal 19	√(9)+ √(9)	sqrt(9)+sqrt(9)	Ya
Soal 20	SIN(10)	sin(10)	Ya

Dari uji coba pada pengenalan karakter diketahui bahwa dari 20 soal yang diujikan, sebanyak 17 gambar soal dapat dikenali dengan benar. Sedangkan 3 diantaranya terjadi kesalahan karena angka 0 dianggap sama sebagai huruf O. Dari hasil ini menunjukkan bahwa tingkat keakuratan proses pengenalan karakter dengan metode *template* matching correlation ini cukup akurat yakni sebesar 85%.

5. Penutup

5.1. Kesimpulan

Dari penelitian yang dilakukan dapat diambil beberapa kesimpulan antara lain :

1. Faktor pencahayaan sangat berpengaruh terhadap kualitas preprosessing yang dihasilkan.
2. Sudut pengambilan gambar sangat berpengaruh terhadap hasil proses segmentasi.
3. Metode *Template* Matching memberikan hasil yang cukup akurat untuk pengenalan karakter.

Penelitian yang telah di lakukan masih memiliki kekurangan antara lain:

1. Cukup sulit membedakan huruf O dan angka 0.
2. Gambar yang diambil dengan kemiringan tertentu akan mengalami kesalahan saat dilakukan pengenalan karakter

5.2. Saran

Dari kesimpulan dan kekurangan sistem diatas, maka untuk perbaikan dan peningkatan lebih lanjut disarankan

1. Menambahkan metode yang dapat lebih baik dalam membedakan huruf O dan angka 0.
2. Menambahkan metode yang dapat menrotasi atau mendeteksi teks yang diambil dengan kemiringan tertentu.

6. Daftar Pustaka

[1] S. K. Singla and R. K. Yadav, “Optical character recognition based speech synthesis system using LabVIEW,” *Journal of Applied Research and Technology*, vol. 12, no. 5, pp. 919–926, 2014.

[2] K. Hamad and M. Kaya, "A Detailed Analysis of Optical Character Recognition Technology," *International Journal of Applied Mathematics, Electronics and Computers*, vol. 4, no. Special Issue-1, pp. 244–244, 2016.

[3] K. Apriyanti and T. Wahyu Widodo, "Implementasi Optical Character Recognition Berbasis Backpropagation untuk Text to Speech Perangkat Android," *IJEIS (Indonesian Journal of Electronics and Instrumentation Systems)*, vol. 6, no. 1, p. 13, 2016.

[4] N. Sahu and M. Sonkusare, "A Study on Optical Character Recognition Techniques," *International Journal of Computational Science, Information Technology and Control Engineering*, vol. 4, no. 1, pp. 01–15, 2017.

[5] G. Baek and S. Kim, "Two Step *Template* Matching Method with Correlation Coefficient and Genetic Algorithm," in *Huang DS., Jo KH., Lee HH., Kang HJ., Bevilacqua V. (eds) Emerging Intelligent Computing Technology and Applications. With Aspects of Artificial Intelligence. ICIC 2009.*, Berlin, Heidelberg: Springer, 2009.

[6] F. A. Hermawati and R. Koesdijarto, "A Real-Time License Plate Detection System," *TELKOMNIKA: Indonesian Journal of Electrical Engineering*, vol. 8, no. 1, pp. 97–106, 2010.

[7] S. A. Dongare, D. B. Kshirsagar, N. J. Khapale, and A. D. Pawar, "Artificial Neural Network For Recognition Of Handwritten Devanagari Character," *IOSR Journal of Computer Engineering (IOSR-JCE)*, vol. 17, no. 1, pp. 60–64, 2015.

[8] R. Anugrah and K. B. Y. Bintoro, "Latin Letters Recognition Using Optical Character Recognition to Convert Printed Media Into Digital Format," *Jurnal Elektronika dan Telekomunikasi*, vol. 17, no. 2, p. 56, 2017.

[9] S. P. Raghavendra and A. Danti, "A novel recognition of Indian bank cheques using feed forward neural network," *IOSR Journal of Computer Engineering (IOSR-JCE)*, vol. 20, no. 3, pp. 44–59, 2018.

[10] F. A. Hermawati and N. Sholeh, "Pengenalan Rambu Batas Kecepatan Pada Sebuah Citra Dengan Metode *Template* Matching," *KONVERGENSI*, vol. 6, no. 1, pp. 1–8, 2010.

[11] G. Baek and S. Kim, "Two step *template* matching method with correlation coefficient and genetic algorithm," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5755 LNAI, pp. 85–90, 2009.