BAB II
LANDASAN TEORI

2.1. Pengenalan Pola

Pengenalan pola adalah mengelompokkan data numerik dan simbolik (termasuk citra) secara otomatis oleh mesin (komputer). Tujuan pengelompokkan adalah untuk pengenalan pola suatu objek di dalam citra. Komputer menerima masukan berupa citra objek yang akan diidentifikasi, memproses citra tersebut dan memberikan keluaran berupa informasi/deskripsi objek di dalam citra [6].

Proses ini mempunyai data masukan dan informasi keluaran yang berbentuk citra. Operasi pengolahan citra digital umumnya dilakukan dengan tujuan memperbaiki kualitas suatu gambar sehingga dapat dengan mudah diinterpretasikan oleh mata manusia dan untuk mengolah informasi yang terdapat pada suatu gambar untuk keperluan pengenalan objek secara otomatis.

2.1.1. Konversi RGB ke *Grayscale*

Dalam pengaturan RGB, setiap *pixel* memiliki tiga bagian bayangan: Merah, Hijau, dan Biru. Pra-persiapan, gambar bayangan diberikan sebagai info dan diubah menjadi gambar skala gelap. Langkah awal untuk mendigitalkan gambar "sangat kontras" melalui berbagai nuansa redup adalah *mengisolasi* gambar ke dalam berbagai *pixel*, bergantung pada penentuan spasial yang diperlukan. Rentang ini diucapkan dalam jalur dinamis sebagai rentang dari 0 (gelap) dan 1 (putih), dengan nilai-nilai fragmentaris [7].

2.1.2. *Edge Detection* menggunakan Metode *Robert*

Metode *robert* merupakan metode yang menggunakan operator *robert*. Operator *robert* merupakan operator yang berbasis gradient yang menggunakan dua buah *kernel* yang berukuran 2x2 piksel. Operator ini mengambil arah diagonal untuk penentuan arah dalam penghitungan nilai gradien, sehingga sering disebut dengan operator silang [8].

Perhitungan gradien pada operator *robert* adalah sebagai berikut ini:

$$G = \sqrt{R_x^2 + R_y^2} \quad \text{……..……..... (1)}$$

Dimana,

G = besar gradient operator *robert*

Rx = gradien *robert* arah horizontal

Ry = gradien *robert* arah vertikal

Kernel yang digunakan pada konvolusi Rx dan Ry adalah 2x2 dan kemudian dihitung, contohnya sebagai berikut ini:

$$R_x = \begin{matrix} 1 & 0 \\ 0 & -1 \end{matrix} \qquad R_y = \begin{matrix} 0 & 1 \\ -1 & 0 \end{matrix}$$

Cara kerja dari metode *robert* ini untuk menemukan *edge* adalah dengan menggunakan pendekatan *robert* pada derivativenya. Dan hasilnya akan mengembalikan nilai *edge* pada titik-titik tersebut yang dimana gradien dari citra adalah maksimum. Pada metode ini hanya akan mendeteksi *edge* yang kuat dari *threshold* yang telah ditentukan, apabila tidak ditentukan *threshold* nya maka dari matlabnya akan memilih nilainya secara *default*.

2.1.3. Segmentasi Karakter

Tujuan dari tahap ini, mengingat gambar perluasan adalah untuk menyegmentasikan setiap karakter tanpa kehilangan komponen karakter. Salah satu yang paling menonjol di antara prosedur yang paling penting penghargaan plat nomor yang diprogram. Di luar kemungkinan bahwa pembagian itu menjadi datar, karakter dapat di *isolasi* menjadi dua bagian, atau dua karakter bisa di *konsolidasi*. Dengan mengingat tujuan akhir untuk melihat karakter plat nomor kendaraan beberapa saat kemudian, setiap karakter harus dipartisi secara individual. Karakter individu harus dikenali (dipotong) dari satu sama lain.

Dalam segmentasi karakter, karakter dan digit dari plat dibagian dan masing-masing terhindar sebagai berbagai gambar. Ini mengkuantifikasi pengaturan properti untuk setiap distrik yang ditandai dalam kisi nama. Kotak memantul digunakan untuk mengukur properti distrik gambar. Sistem ini

digunakan untuk memeriksa angka-angka dengan format yang digunakan oleh perhitungan tata letak koordinat dalam *Optical Character Recognition* (*OCR*) [9]. Segmentasi karakter pada pengenalan plat nomor motor yang akan dibuat menggunakan *tools* Matlab.

2.2. *Optical Character Recognition* (OCR) Menggunakan OCR *Algorithm*

Optical Character Recognition (OCR) adalah sebuah aplikasi komputer yang digunakan untuk mengidentifikasi citra huruf maupun angka untuk dikonversi ke dalam bentuk file yang dapat dikenali oleh komputer [10]. Pencocokan *template* adalah salah satu prosedur *Character Acknowledgement*. Ini adalah cara untuk menemukan *area sub*-gambar yang disebut *format* di dalam gambar. Koordinasi *format* termasuk menentukan kesamaan antara *template* yang diberikan dan jendela dengan ukuran yang sama dalam gambar dan mengenali jendela yang memberikan ukuran kemiripan yang paling penting. Bekerja dengan korelasi *pixel-by-pixel* dari gambar dan *format* untuk setiap pencabutan tata letak yang memungkinkan. Prosedur ini termasuk pemanfaatan basis data karakter atau tata letak. Terdapat format untuk semua karakter informasi yang dapat dibayangkan. Format dibuat untuk masing-masing karakter alfanumerik (dari A-Z dan 0-9) menggunakan gaya-gaya teks "Normal" [11]. Adapun *OCR algoritm* yang digunakan adalah *Optical Character Recognition* (*OCR*) menggunakan metode *Template Matching*.

Dalam penelitian yang dibuat oleh Md. Anwar Hossain & Sadia Afrin pada jurnal *Optical Character Recognition based on Template Matching*, OCR menyediakan visualisasi alfanumerik dengan memindai gambar teks dan mengubahnya menjadi dokumen teks. Tujuan utama dari prototipe sistem ini adalah untuk mengimplementasikan algoritma *Template Matching*. Dalam penelitian tersebut, karakter yang digunakan adalah alfabet (A-Z dan a-z), dan angka (0-1), gambar skala abu-abu, format gambar *bitmap* digunakan dan mengenali alfabet dan angka dengan membandingkan antara dua gambar. Selain itu, penelitian tersebut memeriksa akurasi berbagai font alfabet dan angka [12].

Dalam penelitian yang dibuat oleh Rachit Virendra Adhvaryu pada jurnal *Optical Character Recognition Using Template Matching* (*Alphabets* & *Numbers*), OCR menjadi salah satu aplikasi teknologi paling sukses di bidang pengenalan pola

dan kecerdasan buatan. OCR menggunakan Template Matching adalah prototipe sistem yang berguna untuk mengenali karakter atau alfabet dengan membandingkan dua gambar alfabet. Prosesnya dimulai dari proses akuisisi, proses *filtering*, *threshold*, pengelompokan gambar alfabet dan akhirnya mengenali alphabet [13].

2.3. Template Matching

Pada dasarnya *template matching* adalah proses yang sederhana. Suatu citra masukan yang mengandung *template* tertentu dibandingkan dengan template pada basis data. *Template* ditempatkan pada pusat bagian citra yang akan dibandingkan dan dihitung seberapa banyak titik yang paling sesuai dengan *template* [14]. Langkah ini diulangi terhadap keseluruhan citra masukan yang akan dibandingkan. Nilai kesesuaian titik yang paling besar antara citra masukan dan citra template menandakan bahwa *template* tersebut merupakan citra *template* yang paling sesuai dengan citra masukan.

Template diposisikan pada citra yang akan dibandingkan dan dihitung derajat kesesuaian pola pada citra masukan dengan pola pada citra *template*. Tingkat kesesuaian antara citra masukan dan citra *template* bisa dihitung berdasarkan nilai error terkecil. *Template* dengan nilai error paling kecil adalah *template* yang paling sesuai dengan citra masukan yang akan dibandingkan. Ukuran objek yang beragam bisa diatasi dengan menggunakan *template* berbagai ukuran. Namun hal ini membutuhkan tambahan ruang penyimpanan. Penambahan *template* dengan berbagai ukuran akan membutuhkan *komputasi* yang besar. Jika suatu *template* berukuran persegi dengan ukuran m x m dan sesuai dengan citra yang berukuran N x N, dan dimisalkan *pixel* m2 sesuai dengan semua titik citra, maka *komputasi* yang harus dilakukan adalah O(N2m2). *Komputasi* tersebut harus dilakukan dengan *template* yang tidak beragam. Jika parameter *template* bertambah, seperti ukuran *template* yang beragam, maka komputasi yang dilakukan juga akan bertambah.

Pengenalan pola dengan menggunakan metode *template matching* dilakukan dengan cara membandingkan citra masukan dengan citra *template*. Citra masukan dihitung berdasarkan banyaknya titik yang sesuai dengan citra template. *Pixel* citra *biner* ditelusuri mulai dari kiri atas hingga ke kanan bawah. Citra *biner* dengan *pixel* berwarna hitam akan direpresentasikan dengan nilai 1. Sedangkan pixel citra yang

berwarna putih akan direpresentasikan dengan nilai 0. Deretan angka biner pada citra masukan akan dihitung dengan deretan angka biner pada citra *template*. *Template* dengan nilai eror terkecil merupakan *template* citra yang paling sesuai dengan citra masukan.

Dalam penelitian yang dibuat oleh Suryo Hartanto, Aris Sugiharto, dan Sukmawati Nur Endah pada jurnal *Optical Character Recognition Menggunakan Algoritma Template Matching Correlation*, OCR merupakan solusi yang efektif untuk proses konversi dokumen cetak ke dokumen digital. Permasalahan yang timbul dalam penelitian tersebut adalah bagaimana teknik pengenalan untuk mengidentifikasi berbagai jenis karakter dengan berbagai ukuran dan bentuk. Metode yang digunakan dalam penelitian tersebut adalah *Template Matching Correlation*. Sebelum proses pengenalan, citra masukan dengan format * bmp atau jpg * diolah terlebih dahulu di proses *preprocessing*, yang meliputi binerisasi, segmentasi, dan normalisasi gambar. Rata-rata tingkat keberhasilan yang dihasilkan oleh penelitian tersebut adalah 92,90%. Hasil akhir menunjukkan bahwa penggunaan metode *Template Matching Correlation* cukup untuk membangun sebuah sistem OCR dengan akurasi yang baik efektif [15].

Dalam penelitian yang dibuat oleh Dr. S.Vijayarani dan Ms. A.Sakila pada jurnal *Template Matching Technique For Searching Words In Document Images*, *Optical Character Recognition* (OCR) digunakan untuk mengubah gambar input yang diberikan ke bentuk teks yaitu bentuk dokumen normal yang dapat melakukan semua tugas pengeditan. Selain itu, alat OCR mengambil teks dari gambar dengan menerapkan teknik *template matching*. *Template matching* adalah teknik yang digunakan dalam pemrosesan gambar digital untuk menemukan bagian kecil dari suatu gambar yang cocok dengan gambar template. *Template Matching* diterapkan untuk gambar dokumen pindaian yang berisi karakter (huruf besar dan kecil) dan angka. Untuk melakukan perbandingan gambar template dengan gambar input, menggunakan metode Indeks Kinerja dan dibandingkan dengan normalisasi korelasi silang dan metode korelasi silang [16].

2.4. Matlab

Matlab (*mathematics laboratory*) merupakan bahasa pemrograman yang digunakan untuk mengerjakan operasi matematika atau operasi aljabar matriks. Untuk pengembangan algoritma, matlab menyediakan antarmuka command line, sebuah fungsi manipulasi string dan bilangan, 2D dan 3D plooting *function* dan kemampuan untuk membuat tampilan GUI (*graphical user interface*) [17].

GUIDE atau GUI *builder* merupakan sebuah *graphical user interface* (GUI) yang dibangun dengan obyek grafis seperti tombol (*button*), kotak teks, *slider*, sumbu (*axes*), maupun menu. Aplikasi yang menggunakan GUI umumnya lebih mudah dipelajari dan digunakan karena orang yang menjalankannya tidak perlu mengetahui perintah yang ada dan bagaimana perintah bekerja.

Tidak seperti bahasa pemrograman lainnya, GUIDE matlab memiliki banyak keunggulan tersendiri antara lain:

1. GUIDE matlab banyak digunakan dan cocok untuk aplikasi-aplikasi berorientasi sains, sehingga banyak peneliti atau mahasiswa menggunakan GUIDE matlab.
2. Matlab memiliki banyak fungsi *built in* yang siap digunakan dan tidak perlu repot membuatnya sendiri.
3. Ukuran file, baik FIG-file maupun M-file, yang dihasilkan relatif kecil.
4. Kemampuan grafisnya cukup andal dan tidak kalah dibandingkan bahasa pemrograman lainnya [18].

2.5. Python

Python merupakan bahasa pemrograman yang berorientasi obyek dinamis, dapat digunakan untuk bermacam macam pengembangan perangkat lunak. Python menyediakan dukungan yang kuat untuk integrasi dengan bahasa pemrograman lain dan alat-alat bantu lainnya. Python hadir dengan pustaka-pustaka standar yang dapat diperluas serta dapat dipelajari hanya dalam beberapa hari. Bahasa pemrograman yang interpretatif multiguna dengan filosofi perancangan yang berfokus pada tingkat keterbacaan kode. Python diklaim sebagai bahasa yang menggabungkan kapabilitas, kemampuan, dengan sintaksis kode yang sangat jelas,

dan dilengkapi dengan fungsionalitas pustaka standar yang besar serta komprehensif [19].

Beberapa fitur yang dimiliki python adalah:

1. Memiliki aturan *layout* kode sumber yang memudahkan pengecekan, pembacaan kembali dan penulisan ulang kode sumber.
2. Memiliki sistem pengelolaan memori otomatis (garbage collection, seperti java) sehingga dapat menghindari pencatatan kode.
3. Pemrograman berorientasi objek.
4. Modular, mudah dikembangkan dengan menciptakan modul-modul baru. Modul-modul tersebut dapat dibangun dengan bahasa Python maupun C/C++.

2.6. Google Cloud

Google Cloud Vision API memungkinkan untuk memahami konten gambar dengan merangkum model pembelajaran mesin yang kuat dalam REST API dan mudah digunakan. Dengan cepat mengklasifikasikan gambar ke dalam ribuan kategori, mendeteksi objek dan wajah individual dalam gambar, dan menemukan serta membaca kata-kata tercetak yang terkandung dalam gambar[20].

Google Cloud Vision API mengambil model pembelajaran mesin yang sangat kompleks yang berpusat di sekitar pengenalan gambar dan memformatnya dalam antarmuka API REST yang sederhana. Ini mencakup berbagai pilihan alat pengenalan gambar dan dapat menggunakannya untuk aplikasi dunia nyata seperti mengategorikan gambar atau memoderasi konten yang menyinggung.