# BAB I PENDAHULUAN

1. **Latar Belakang**

Industri perawatan kulit (*skincare*) telah mengalami perkembangan yang sangat pesat dalam beberapa tahun terakhir. Pertumbuhan ini terlihat dari meningkatnya kesadaran masyarakat akan pentingnya perawatan diri dan perhatian terhadap kesehatan kulit (Situmeang dan Claretta, 2024). *Skincare* kini telah menjadi bagian penting dalam rutinitas harian banyak individu. ‘masker, dirancang untuk mendukung kesehatan serta kecantikan kulit, sekaligus mengatasi berbagai masalah kulit seperti jerawat dan penuaan (Pokhrel, 2024).

Meskipun semakin beragam produk yang beredar, seringkali orang kesulitan dalam memahami informasi kandungan bahan yang tertera pada label produk. Label produk *skincare* sering kali mencantumkan istilah asing yang sulit dimengerti oleh masyarakat awam. Kesulitan ini tidak hanya dipengaruhi oleh kompleksitas istilah yang digunakan, tetapi juga oleh minimnya pemahaman mengenai kandungan dalam produk skincare itu sendiri. Akibatnya, banyak konsumen yang kurang memahami manfaat, cara penggunaan, serta potensi efek samping dari bahan-bahan yang terdapat dalam produk yang individu pilih. Menurut Akbar (2025) Kurangnya literasi mengenai kandungan bahan aktif, penggunaan yang benar, serta kemungkinan efek samping dari produk *skincare* dan kosmetik dapat meningkatkan risiko gangguan kulit, seperti alergi, iritasi, hingga kerusakan dalam jangka panjang. Kondisi ini disebabkan oleh belum tersedianya aplikasi atau platform yang mampu menampilkan informasi kandungan *skincare* secara menyeluruh dengan cepat dan akurat. Oleh karena itu, diperlukan sarana yang dapat membantu masyarakat memahami kandungan produk dengan lebih mudah dan tepat guna, sehingga individu dapat memilih produk yang sesuai dengan kebutuhan kulitnya.

Untuk mengatasi masalah tersebut, teknologi *Optical Character Recognition (OCR)* dapat menjadi solusi yang efektif. Menurut Gunawan et al. (2014), OCR pada dasarnya ialah pengenalan karakter *alphanumeric* dari karakter tulisan tangan atau file maupun citra menjadi teks yang dapat diedit. Metode ini sangat berguna untuk mendeteksi dan menampilkan kandungan bahan yang tercantum pada label produk skincare secara otomatis dan akurat. Dengan memanfaatkan OCR, aplikasi dapat memindai gambar label produk, mengidentifikasi bahan-bahan yang terkandung, dan memberikan informasi terkait manfaat dan potensi risiko bahan tersebut bagi pengguna.

Beberapa penelitian sebelumnya telah mengkaji penerapan OCR dalam berbagai bidang. Penelitian oleh Gunawan et al. (2014), menunjukkan terealisasinya program deteksi dan membaca karakter plat nomor kendaraan yang diambil secara langsung menggunakan kamera webcam. Penelitian tersebut menggunakan metode OCR dengan tingkat keberhasilan pada berbagai jarak yaitu 150 cm sebesar 93,3%, pada jarak 200 cm sebesar 80%, dan pada jarak 250 cm sebesar 46,6%. Penelitian lainnya dilakukan oleh Sanjaya (2021), yang mengimplementasikan metode OCR menggunakan *String Matching* untuk mendeteksi obat dan makanan berbasis *android*. Penelitian tersebut menggunakan algoritma *neural network backpropagation* yang memiliki ketepatan dan kecocokan pendeteksian yang baik dari 7 sample dengan persentase keberhasilan pendeteksian sebesar 96%.

Berdasarkan uraian di atas dan penelitian terdahulu, maka penulis tertarik melakukan penelitian tentang Identifikasi Kandungan Bahan Produk *Skincare* menggunakan Metode Optical Character Recognition (OCR). Dengan adanya aplikasi ini, diharapkan dapat mempermudah konsumen dalam memilih produk yang sesuai dengan kebutuhan dan kondisi kulit mereka, sekaligus meningkatkan kesadaran akan pentingnya memahami komposisi bahan dalam produk *skincare*.

1. **Rumusan masalah**

Berdasarkan latar belakang maka peneliti dapat menyimpulkan rumusan masalah sebagai berikut:

1. Bagaimana mengidentifikasi kandungan bahan yang tertera pada kemasan produk skincare menggunakan metode OCR?
2. Seberapa tinggi tingkat akurasi metode OCR dalam membaca teks pada kemasan produk *skincare*?
3. **Tujuan penelitian**

Adapun tujuan penelitian berdasarkan rumusan masalah di atas adalah sebagai berikut:

1. Mengembangkan aplikasi berbasis Web yang mampu membaca kandungan bahan produk *skincare* menggunakan metode OCR.
2. Mengevaluasi akurasi metode OCR dalam membaca teks pada label produk *skincare*.
3. **Manfaat penelitian**

Penelitian ini diharapkan dapat memberikan manfaat, antara lain:

1. Memberikan kontribusi pada pengembangan teknologi OCR di bidang kosmetik dan skincare, khususnya dalam implementasi aplikasi untuk deteksi kandungan bahan produk.
2. Mempermudah konsumen dalam memahami kandungan bahan produk skincare menggunakan teknologi OCR, sehingga dapat membantu mereka membuat keputusan yang lebih bijak dan tepat.
3. Menjadi referensi bagi penelitian lanjutan terkait pengembangan aplikasi berbasis OCR di bidang kosmetik, terutama untuk meningkatkan transparansi informasi produk dan inovasi teknologi pendukungnya.
4. **Batasan masalah**

Agar mencegah meluasnya permasalahan yang ada, penelitian ini memiliki batasan-batasan sebagai berikut :

1. Dataset yang digunakan terdiri dari 140 gambar kemasan produk skincare dari 20 merek yang terbagi dalam 7 kategori, dan mencakup berbagai bentuk kemasan seperti botol, tube, serta kotak yang relatif datar.
2. Penelitian ini tidak mencakup kondisi kemasan yang rusak, dan buram sehingga memengaruhi akurasi OCR.

# BAB II TINJAUAN PUSTAKA

1. **Produk Skincare**

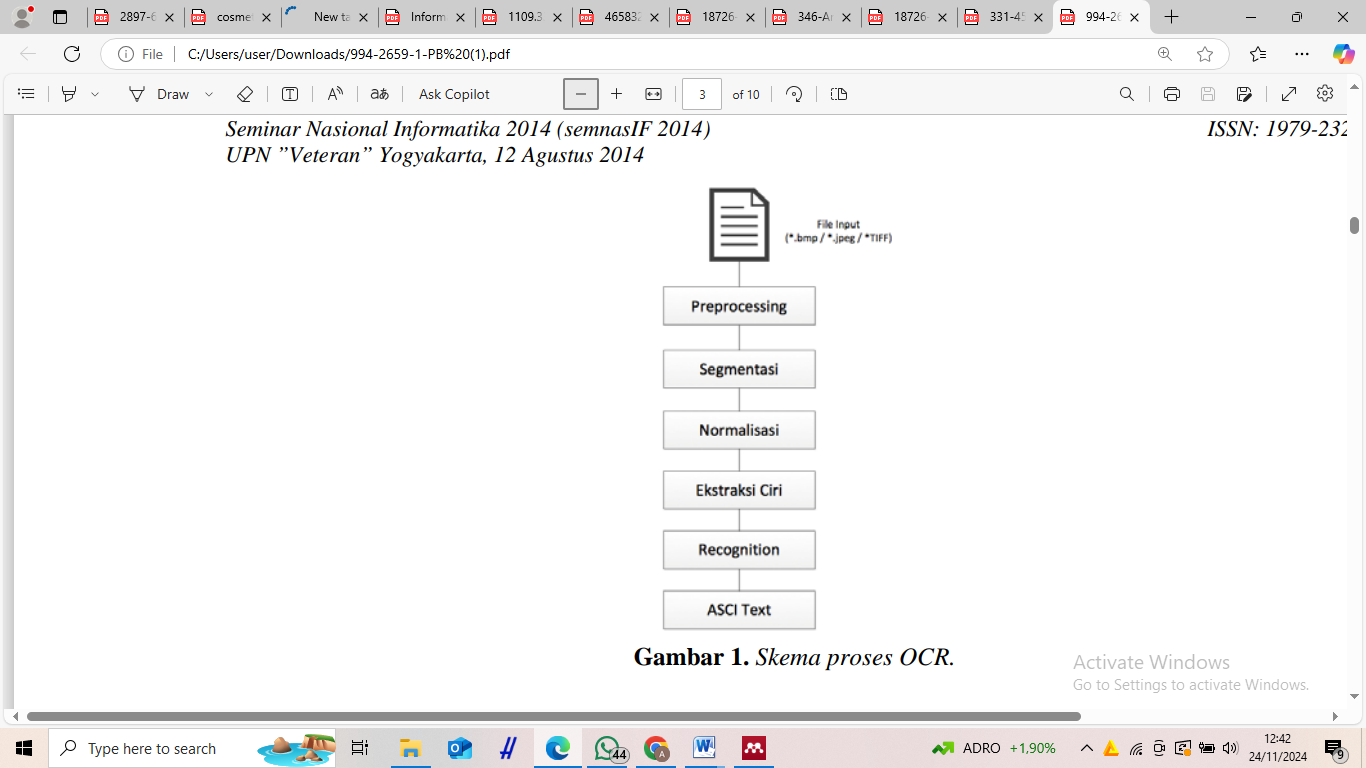
Perawatan kulit atau skincare mengacu pada serangkaian tindakan yang bertujuan untuk menjaga kesehatan kulit, memperbaiki tampilannya, dan mengatasi masalah kulit. Aktivitas ini mencakup pemberian nutrisi, menghindari paparan sinar matahari yang berlebihan, serta penggunaan pelembap secara tepat. Produk skincare biasanya mengandung berbagai bahan aktif yang memiliki manfaat tertentu bagi kulit, seperti antioksidan, senyawa antipenuaan, dan pelembap (De Almeida et al., 2023).

Informasi tentang kandungan bahan dalam produk skincare sangat penting untuk membantu konsumen memahami manfaat produk, menghindari bahan yang dapat memicu alergi atau iritasi, serta memilih produk yang sesuai dengan jenis dan kebutuhan kulit mereka. Menurut Indrisari, (2024) Rendahnya tingkat pengetahuan konsumen dalam memilih produk kosmetik sering kali menyebabkan berbagai masalah kulit, terutama pada wanita, seperti kemerahan, noda hitam, kulit mengelupas, reaksi alergi, hingga munculnya bruntusan. Oleh karena itu, literasi yang baik tentang bahan-bahan dalam produk skincare dapat membantu konsumen membuat keputusan yang lebih bijaksana dan aman dalam memilih produk skincare (Angriani., 2023).

1. **Pengolahan Citra**

Pengolahan citra (*image-processing*) adalah suatu proses pengolahan citra dasar menjadi citra lain yang memiliki kualitas lebih baik dari sebelumnya, pengolahan ini dilakukan dengan cara memanipulasi citra dasar menjadi citra lain dengan menggunakan teknik atau algoritma tertentu. Tujuannya adalah untuk memperbaiki kualitas dari citra agar mudah ditafsirkan oleh manusia maupun komputer, mentransformasikan citra menjadi citra lain, merupakan proses awal dari *computer vision*. Hal-hal yang harus diperhatikan pada proses pengolahan citra digital, diantaranya adalah teknik dalam mengambil citra, proses *filtering*, sampling dan kuantitasi, histogram, perbaikan pada citra serta pengolahan citra digital yang lebih lanjut seperti *image clustering*, segmentasi dan ekstrasi ciri (Sulistiyo & Nugraha Saian, 2019).

1. ***Optical Character Recognition* (OCR)**

*Optical Character Recognition* (OCR) adalah sebuah sistem komputer yang dapat membaca huruf, baik yang berasal dari sebuah pencetak (printer atau mesin ketik) maupun yang berasal dari tulisan tangan (Setiawan et al., 2017).Menurut Sandhika, (2014) OCR adalah sebuah aplikasi komputer yang digunakan untuk mengidentifikasi citra huruf maupun angka untuk dikonversi ke dalam bentuk file tulisan. Sistem pengenal huruf ini dapat meningkatkan fleksibilitas atau kemampuan dan kecerdasan komputer. Sistem pengenal huruf yang cerdas sangat membantu usaha digitalisasi informasi dan pengetahuan, misalnya dalam pembuatan koleksi pustaka digital, koleksi sastra kuno, dan lain – lain. Secara umum proses OCR dapat dilihat pada gambar berikut.

Menurut Pangestu, (2015) sebelum menjadi teks mentah, karakter sebuah citra akan melalui serangkaian proses sehingga dapat dikenali satu-persatu.

* 1. *Preprocessing*

Tahap ini merupakan tahap awal penerjemahan. Dalam *preprocessing*, dilakukan penghilangan *noise* agar pola-pola yang tidak dibutuhkan dihilangkan.

* 1. Segmentasi

Tahap ini merupakan tahap dimana aplikasi menentukan letak teks dalam citra sehingga memudahkan pemilihan wilayah pemindaian. Proses ini akan membagi citra menjadi dua wilayah, yaitu wilayah latar dan wilayah teks. Setelah melakukan pembagian, OCR hanya akan melakukan proses selanjutnya pada wilayah teks yang sudah tersegmentasi.

* 1. Normalisasi

Tahap ini merupakan tahap pengecekan bentuk, pola, maupun ketebalan dari setiap karakter yang terdeteksi. Dimisalkan OCR menerima masukkan berupa sebuah kata yang terdiri dari berbagai ukuran. Pada proses normalisasi akan dideteksi perbedaan tersebut agar membentuk sebuah keseragaman sehingga mempermudah OCR untuk melakukan ekstraksi di tahap selajutnya.

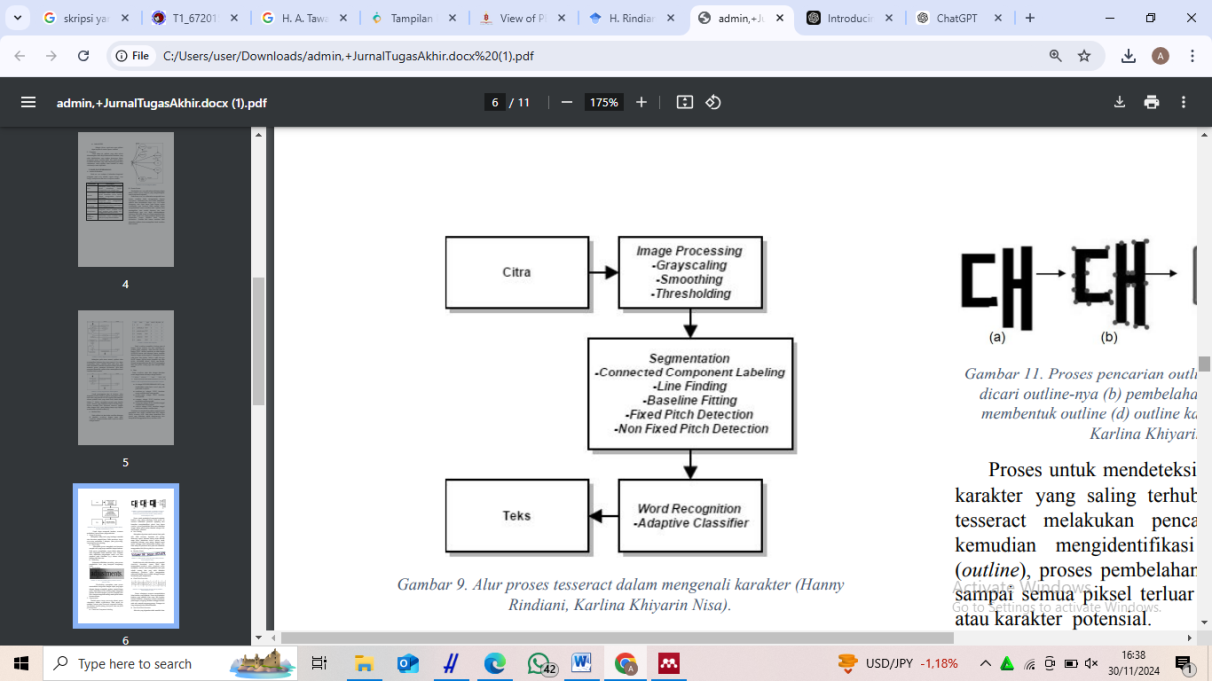
* 1. Ekstraksi

Tahap ini merupakan pengambilan ciri khas dari karakter yang terdeteksi dan ternormalisasi. Hal ini menyebabkan OCR dapat mengetahui *typeface* dari tulisan yang digunakan.

* 1. *Recognition*

Tahap ini merupakan tahap akhir penerjemahan. informasi-informasi yang sudah diambil dari ekstraksi. Karakter yang sudah dipisah satu-persatu tersebut kemudian akan dibandingkan dengan polapola karakter yang bersesuaian dengan pola yang sudah ada, biasanya disimpan di dalam sebuah basis data.

1. ***Tesseract* OCR**

*Tesseract* sendiri sejak 2006 dikembangkan oleh *Google* dengan lisensi Apache 2.0 yang berarti *library* OCR tersebut bebas digunakan (*open source*). *Library* ini dapat digunakan secara langsung, atau (untuk pemrogram) menggunakan API untuk mengekstrak teks yang dicetak dari gambar. *Tesseract* tidak memiliki GUI bawaan, tetapi ada beberapa yang tersedia dari halaman Pihak Ketiga. *Tesseract* kompatibel dengan banyak bahasa pemrograman dan kerangka kerja melalui pembungkus yang dapat ditemukan di sini (Muhtadii & Tawakal, 2016).

Menurut Sulistiyo dan Nugraha Saian, (2019) untuk mengenali karakter, *Tesseract* menjalankan tiga proses utama, yaitu:

1. ***Image Processing***

Tahapan awal ini berfungsi untuk mengubah citra berwarna menjadi biner. Proses ini melibatkan tiga langkah utama yaitu *grayscaling, smoothing, dan thresholding*.

1. *Grayscaling*

Langkah ini bertujuan mengonversi citra berwarna menjadi citra dengan tingkat keabuan saja. Hasil dari proses ini akan digunakan untuk melakukan binerisasi citra. Proses pengubahan dilakukan dengan memindai setiap piksel pada koordinat (x, y) dalam ukuran panjang dan lebar citra.

Citra berwarna terdiri dari warna piksel diwakili oleh kombinasi tiga warna dasar komponen yaitu *red* (r), *green* (g) ​​dan *blue* (b). Kisaran nilai untuk semua komponen warna ini adalah 0-255 (Andersson & Isaksen, 2002). Jadi, itu nilai skala abu-abu yang sesuai 𝑓(𝑥, 𝑦) untuk setiap piksel, yang juga terletak antara 0-255, dapat diperoleh dengan menggunakan persamaan sebagai berikut.

|  |  |
| --- | --- |
|  | (1) |

1. *Smoothing*

Langkah berikutnya adalah *smoothing*, yaitu proses menghaluskan citra untuk mengurangi *noise* atau gangguan pada gambar. Salah satu metode yang umum digunakan untuk *smoothing* adalah *Gaussian Blur*. Menurut Firdaus & Imelda, (2018) Metode *Gaussian Blur* digunakan untuk mengurangi *noise* pada sebuah citra, Untuk menerapkan *Gaussian Blur*, diperlukan proses konvolusi. Konvolusi merupakan operasi penjumlahan dari hasil perkalian antara matriks *filter* dan matriks tetangga di sekitar titik (x,y) pada sebuah citra. Berikut persamaan untuk menghitung *Gaussian Blur.*

|  |  |
| --- | --- |
|  | (2) |

|  |  |
| --- | --- |
|  | = Elemen matriks *Gauss* pada posisi (x, y) |
|  | = *Filter* radius atau standar deviasi |
|  | = Ukuran matriks *Gauss* yang jangkauannya dari sampai dengan titik tengah berapa pada *x=0* dan *y=0*. |

1. *Thresholding*

|  |
| --- |
| Gambar 3. *Thresholding* |

*Thresholding* adalah proses yang digunakan untuk memisahkan latar belakang dari objek yang diamati dengan mengonversi gambar menjadi format hitam-putih. Dalam tahap ini, *thresholding* dapat mengubah gambar berwarna atau *grayscale* menjadi gambar biner dengan menyesuaikan nilai piksel berdasarkan rentang tertentu. Prinsip dasar *thresholding* adalah mengubah intensitas warna piksel menjadi dua nilai, yaitu 0 dan 255. Proses ini membagi piksel menjadi dua kelompok yaitu piksel dengan nilai intensitas 0 hingga 127 akan diubah menjadi 0, sedangkan piksel dengan nilai intensitas 128 hingga 255 akan diubah menjadi 255.

1. ***Segmentation***

Setelah tahap *image processing* selesai, langkah berikutnya adalah *segmentation*. Pada tahap ini, terdapat empat proses utama, yaitu *connected component labeling, line finding, baseline fitting*, serta deteksi karakter dengan *pitch* tetap (*fixed pitch*) dan tidak tetap (*non-fixed pitch*).

* + 1. *Connected Component Labeling*

|  |
| --- |
| Gambar 4. *Connected Component Labeling* |

Merupakan proses untuk mengidentifikasi komponen karakter yang saling terhubung. *Tesseract* akan memindai seluruh citra dan mendeteksi piksel latar depan (*outline*). Pemisahan karakter dilakukan secara bertahap hingga semua piksel luar teridentifikasi sebagai *blob* atau kandidat karakter potensial.

* + 1. *Line Finding*

Langkah ini menggunakan algoritma untuk mendeteksi baris teks. Dalam *Tesseract*, algoritma ini dirancang untuk mengenali halaman teks yang miring tanpa perlu melakukan proses *deskew* (penyesuaian kemiringan halaman), sehingga kualitas citra tetap terjaga. Proses ini melibatkan teknik *blob filtering* dan *line construction* untuk menemukan baris pada teks.

* + 1. *Baseline fitting*

|  |
| --- |
| Gambar 5. *baseline fitting* |

Setelah baris teks berhasil diidentifikasi, *baseline* (garis pangkal) disesuaikan dengan lebih akurat menggunakan metode *quadratic spline*. *Quadratic* *spline* adalah teknik yang digunakan untuk menghasilkan titik-titik pada rentang data yang telah ditentukan sebelumnya. Metode ini memanfaatkan polinomial berderajat rendah untuk membentuk garis yang halus dan presisi.

* + 1. *Fixed Pitch Detection*

|  |
| --- |
| Gambar 6. *Fixed Pitch Detection* |

Tahap berikutnya, *Tesseract* memperkirakan lebar karakter yang terdeteksi. Proses ini bertujuan untuk mengenali karakter dengan lebar tetap. Jika karakter berhasil diidentifikasi, *Tesseract* akan melakukan pemotongan (*chopping*) pada karakter tersebut, sehingga setiap karakter dalam teks terpisah menjadi bagian-bagian kecil. Potongan-potongan ini kemudian akan diproses lebih lanjut untuk diklasifikasikan.

* + 1. *Non Fixed Pitch Detection*

Jika teks yang digunakan tidak memiliki lebar garis tepi yang tetap, Tesseract akan menerapkan algoritma *non-fixed pitch detection.* Algoritma ini bekerja dengan mengukur jarak antara *baseline* dan *mean line* (garis tengah). Area pada citra yang mendekati nilai ambang batas (*threshold*) akan dianalisis menggunakan metode *fuzzy* untuk menentukan apakah area tersebut termasuk sebagai fitur yang dapat dikenali atau tidak.

* + 1. *Chopping Characters*

|  |
| --- |
| Gambar 7. *Chopping Characters* |

Pada tahap ini, kandidat titik-titik pemisahan ditentukan berdasarkan simpul cekung, garis poligonal, dan titik cekung yang berlawanan. Pada Gambar 14 di atas, ditunjukkan kumpulan kandidat titik pemisahan yang ditandai dengan panah, sementara garis pemotongan yang dipilih terlihat melintasi *outline* tempat huruf 'r' bersentuhan dengan huruf 'm'. Pemotongan dilakukan berdasarkan urutan prioritas tertentu. Jika pemotongan tidak berhasil, hasil tersebut tidak sepenuhnya diabaikan, melainkan disimpan oleh *associator* agar dapat digunakan kembali jika diperlukan.

* + 1. *Associating Broken Characters*

|  |
| --- |
| Gambar 8. *Associating Broken Characters* |

Jika tahap pemotongan karakter tidak menghasilkan bentuk karakter yang memadai, proses dilanjutkan ke tahap *associator*. Pada tahap ini, *associator* akan melakukan pencarian dengan mengambil kandidat *state* baru dari antrean prioritas. Kandidat tersebut kemudian dievaluasi dengan cara mengklasifikasikan kombinasi fragmen yang sebelumnya belum terklasifikasi.

1. ***Word Recognition***

|  |
| --- |
| Gambar 9. *Word Recognition* |

Setelah proses segmentasi selesai, langkah berikutnya adalah pengenalan kata (*word recognition*). Pada tahap ini, dilakukan identifikasi bentuk karakter melalui proses *adaptive classifier*. Proses ini mencakup pengenalan elemen-elemen poligon yang dipecah menjadi segmen pendek dengan panjang yang seragam, sehingga dimensi panjang dihilangkan dari vektor fitur. Setiap segmen pendek kemudian dibandingkan dengan fitur prototipikal yang telah dilatih sebelumnya, menjadikan proses klasifikasi lebih tangguh terhadap karakter yang terputus.

1. **Aplikasi berbasis Web**

Aplikasi adalah bagian perangkat lunak komputer yang dibuat dengan program komputer untuk digunakan melakukan suatu tugas yang diinginkan oleh pengguna (Moh. Fauzi, 2018:12 dalam Novria Rahma et al., 2022). Aplikasi berbasis Web adalah aplikasi yang dikembangkan dengan menggunakan bahasa pemrograman *HTML, PHP, CSS, JS* yang membutuhkan *web server* dan *browser* untuk menjalankannya seperti *Chrome, Firefox* atau *Opera*, *internet eksplorer*, *Microsoft edge* dan lain-lainnya. Aplikasi ini dapat berjalan pada jaringan maupun internet (Jaringan LAN). Data yang terpusat dan kemudahan dalam mengakses adalah ciri utama yang membuat Aplikasi *Web* lebih banyak diminati dan lebih mudah diimplementasikan di berbagai bidang kehidupan (Janner Simamarta, dkk 2021:113 dalam Novria Rahma et al., 2022).

1. **Python**

Python adalah bahasa pemrograman tingkat tinggi yang dirancang untuk mendukung berbagai kebutuhan pengembangan aplikasi. Dirilis pertama kali pada 1990-an dan sekarang digunakan untuk membuat jutaan aplikasi, permainan, hingga situs web. *Python* merupakan bahasa pemrograman komputer berbasis teks. Dengan penulisan menggunakan kata-kata dalam bahasa Inggris, tanda baca, angka dan simbol. Hal ini membuat kode *Python* menjadi lebih sederhana untuk ditulis, dibaca, dan dipahami (Alif Ramadhan et al., 2024). *Python* menyediakan berbagai *library* yang mendukung beragam kebutuhan, termasuk pemrosesan data, visualisasi, analisis statistik, pemrosesan bahasa alami, pengolahan gambar, dan lainnya (Dirgantara et al., 2014).

1. **Django**

Django adalah *framework full-stack* berbasis *Python* yang dirancang untuk mempermudah pengembangan aplikasi web. Dengan *Django*, proses pembuatan web menjadi lebih cepat karena tidak perlu menulis kode dari awal. Sebagai *framework full-stack*, Django mencakup pengelolaan *front-end* dan *back-end*. *Front-end* mengacu pada bagian antarmuka yang dilihat pengguna, sementara *back-end* menangani pengelolaan *database* dan logika bisnis. *Django* merupakan *framework* tingkat tinggi yang mendorong pengembangan cepat serta menerapkan desain yang bersih dan pragmatis (Dirgantara et al., 2014). Salah satu keunggulan dari *framework* django yakni proses pengembangan aplikasi lebih cepat dari *framework* lain dan merupakan salah satu *framework* paling multifungsi yang dipergunakan oleh banyak sekali organisasi (Alif Ramadhan et al., 2024).

1. ***MySQL***

*MySQL* adalah sistem *database* yang populer untuk pengembangan aplikasi web. Hal ini mungkin disebabkan oleh sifatnya yang gratis, kemudahan dalam pengelolaan data, tingkat keamanan yang baik, aksesibilitas yang tinggi, dan keunggulan lainnya. *MySQL* merupakan sebuah program *database* server yang mampu menerima dan mengirimkan datanya dengan sangat cepat, multiuser serta menggunakan perintah standar *SQL* (*Structured Query Languange*) (Alit et al., 2020). *MySQL* digunakan untuk menyimpan berbagai jenis data dalam *database*, yang dapat diolah sesuai kebutuhan. Pengolahan data tersebut mencakup penambahan, pengeditan, dan penghapusan data yang tersimpan di dalam basis data (Bahri, 2020).

1. **Akurasi**

Akurasi adalah salah satu metrik evaluasi yang digunakan untuk mengukur **kinerja** atau **keandalan** suatu model atau sistem dalam **menghasilkan prediksi yang benar.** Pengujian akurasi merupakan proses yang dilakukan untuk mengevaluasi sejauh mana metode analisis yang digunakan dapat menghasilkan nilai pemulihan (*recovery*) yang optimal. Nilai pemulihan yang diperoleh mencerminkan tingkat kesesuaian hasil analisis dengan kadar analit yang sebenarnya (Ravisankar et al, 2015 dalam Ramadhan & Musfiroh, 2021). Akurasi dihitung dengan cara membandingkan jumlah hasil yang benar dengan jumlah total prediksi yang dilakukan oleh model.

Menurut Brillian & Agustin, (2024) Menentukan tingkat akurasi dari kata-kata yang berhasil diprediksi menggunakan rumus akurasi berikut:

|  |  |
| --- | --- |
| Akurasi x 100% | (3) |

* 1. **Kerangka Pikir**

Identifikasi Masalah

Studi Literatur

Pengumpulan Data

Implementasi Metode OCR

Perancangan Sistem

Tidak

Pengujian Sistem

Ya

Kesimpulan

**BAB III**

**METODE PENELITIAN**

1. **Lokasi dan Tempat Penelitian**

Penelitian ini berlokasi di Laboratorium Terapan Program Studi Matematika, Jurusan Matematika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Tadulako. Pengumpulan dataset berupa gambar kemasan produk *skincare* dilakukan di beberapa toko kosmetik yang ada di Kota Palu.

1. **Alat dan Bahan**

Alat yang digunakan dalam penenlitian ini yaitu laptop Lenovo ideapad 320 dengan spesifikasi *Intel(R) Core(TM) i3-6006U CPU @ 2.00GHz 1.99 GHz*, sistem operasi windows 10, *python* 3.11, *Visual Studio Code*, *Django*, *MySQL*, smarphone dan alat tulis menulis.

1. **Sumber dan Jenis Data**

Sumber data yang digunakan dalam penelitian ini merupakan data primer yang diperoleh langsung melalui pengambilan gambar kemasan produk *skincare* di toko kosmetik yang ada di Kota Palu. Jenis data yang digunakan adalah gambar dengan format JPG atau PNG, yang diambil secara manual menggunakan kamera smartphone. Jenis data yang digunakan adalah gambar kemasan produk yang mengandung teks bahan *skincare*.

1. **Prosedur Penelitian**

Prosedur dalam penelitian ini adalah sebagai berikut :

1. Mulai penelitian
2. Identifikasi masalah

Mengidentifikasi kendala yang sering dialami konsumen dalam memahami kandungan bahan pada kemasan produk *skincare*, terutama yang mencantumkan istilah asing.

1. Studi literatur

Mempelajari metode *Optical Character Recognition* (OCR) serta cara kerja *library Tesseract* untuk mendeteksi teks pada gambar.

1. Pengumpulan data

Data yang digunakan dalam penelitian ini berupa gambar kemasan produk *skincare* dari 20 merek, yang terbagi dalam 7 kategori dan mencakup berbagai bentuk kemasan, seperti botol, *tube* dan kotak yang relatif datar, dengan total 140 gambar. Dataset ini dikumpulkan secara langsung melalui pengambilan gambar manual menggunakan kamera smartphone dalam format JPG atau PNG. Proses pengambilan gambar dilakukan di toko-toko kosmetik yang berlokasi di Kota Palu.

1. Implementasi metode OCR
2. Melakukan tahapan *preprocessing*, segmentasi dan normalisasi gambar kemasan produk *skincare* untuk mempersiapkan gambar sebelum dilakukan deteksi teks menggunakan *tesseract* OCR.
3. Menggunakan *Tesseract* OCR untuk mendeteksi dan mengekstrak teks yang terdapat pada gambar kemasan produk.
4. Hasil deteksi teks kemudian dicocokkan dengan *database* kandungan bahan *skincare* yang telah dibuat untuk memastikan kecocokan antara bahan yang terdeteksi dan daftar bahan yang ada.
5. Perancangan Sistem

Perancangan sistem dilakukan dengan menentukan fitur-fitur utama yang dibutuhkan dalam aplikasi deteksi kandungan bahan *skincare* berbasis web, seperti fitur *upload* gambar dan *scan* gambar kemasan produk. Setelah gambar diproses, sistem menggunakan *Tesseract* OCR untuk mengekstrak teks dari gambar dan mencocokkannya dengan *database* kandungan bahan. Sistem ini juga mencakup pengelolaan *database* untuk menyimpan informasi bahan dan manfaatnya serta antarmuka pengguna yang mudah digunakan. Implementasi sistem dilakukan berdasarkan desain yang telah dibuat untuk memastikan fungsionalitas yang optimal.

1. Pengujian Sistem

Dilakukan evaluasi terhadap kinerja aplikasi yang telah dibangun. Pengujian mencakup berbagai aspek, mulai dari akurasi deteksi teks menggunakan *Tesseract* OCR, kecepatan pemrosesan gambar, hingga kecocokan hasil ekstraksi dengan *database* kandungan bahan *skincare*. Uji coba dilakukan dengan menggunakan gambar kemasan dari berbagai merek dan jenis produk untuk memastikan sistem dapat bekerja dengan baik di berbagai kondisi. Selain itu, pengujian juga dilakukan untuk menilai keandalan fitur *upload* dan *scan* gambar, serta kemudahan penggunaan antarmuka. Hasil dari pengujian ini akan digunakan untuk melakukan perbaikan dan penyempurnaan sistem agar dapat memberikan hasil yang lebih akurat dan efisien.

1. Kesimpulan

Menyusun kesimpulan berdasarkan hasil penelitian.

1. Selesai.
2. **Analisis Data**
3. *Preprocessing*

*Preprocessing* adalah tahap pertama yang bertujuan untuk mempersiapkan gambar kemasan produk skincare agar dapat diproses dengan baik oleh sistem OCR. Pada tahap ini, dilakukan beberapa langkah penting, termasuk:

1. Konversi Gambar ke *Grayscale*

Gambar kemasan produk diubah menjadi format *grayscale* (skala abu-abu) untuk mengurangi kompleksitas dan meningkatkan kontras antara teks dan latar belakang.

1. Penghapusan *Noise*

*Noise* yang ada dalam gambar, seperti titik atau gangguan visual lainnya, dihapus dengan menggunakan filter *Gaussian Blur* untuk membersihkan gambar.

1. *Thresholding*

*Thresholding* adalah proses konversi gambar *grayscale* menjadi gambar biner (hitam-putih). Nilai ambang (*threshold*) diterapkan untuk memisahkan teks dari latar belakang. Bagian gambar dengan intensitas lebih tinggi dianggap sebagai teks (putih), sedangkan latar belakang dianggap sebagai area gelap (hitam).

1. Segmentasi

Pada tahap segmentasi, gambar dibagi menjadi dua wilayah utama, yaitu wilayah teks dan latar belakang. Gambar yang telah diproses pada tahap *preprocessing* kemudian dipisahkan untuk memisahkan teks dari latar belakang.

1. Normalisasi

Setelah teks tersegmentasi, normalisasi dilakukan untuk memastikan bahwa karakter-karakter yang terdeteksi memiliki konsistensi bentuk dan ukuran. Proses ini meliputi skalasi dan penyesuaian ukuran karakter, di mana karakter yang memiliki ukuran atau ketebalan yang berbeda disesuaikan untuk mencapai ukuran yang seragam, sehingga memudahkan proses ekstraksi oleh *tesseract* OCR.

1. Ekstraksi Teks

Pada tahap ekstraksi teks, *Tesseract* OCR digunakan untuk mengekstraksi karakter-karakter yang telah dinormalisasi. Proses ini meliputi identifikasi dan pengolahan karakter, di mana setiap karakter dalam wilayah teks yang tersegmentasi diekstraksi untuk mengenali *font* dan gaya tulisan yang digunakan pada kemasan produk *skincare*.

1. *Recognition* (Pengenalan Teks)

Pada tahap *recognition*, karakter atau kata yang telah diekstraksi dari gambar akan dicocokkan dengan *database* yang berisi pola-pola karakter yang dikenal. Proses ini melibatkan pencocokan karakter yang terdeteksi dengan data yang ada di dalam *database* untuk mengidentifikasi kata atau bahan tertentu. Jika terdapat kecocokan, hasilnya akan mengungkapkan kandungan bahan yang ditemukan pada kemasan produk *skincare* beserta manfaat masing-masing bahan tersebut.

1. Akurasi

Setelah tahap *Recognition*, langkah selanjutnya adalah mengevaluasi sejauh mana sistem OCR berhasil mengenali karakter atau kata yang ada pada gambar kemasan produk *skincare*. Akurasi dihitung dengan membandingkan jumlah hasil yang benar dengan jumlah total prediksi yang dilakukan oleh **sistem OCR**.

* 1. **Sistematika Penulisan**

Untuk memudahkan dan teraturnya penyusunan laporan ini, maka penulisan menyusun secara garis besar dalam 5 (lima) bab yaitu sebagai berikut:

**BAB I : Pendahuluan**

pada bab ini membahas latar belakang, perumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah.

**BAB II : Tijauan Pustaka**

Pada bagian ini berisi teori-teori informasi dan kerangka pikir dalam menunjang penelitan

**BAB III : Metode Penelitian**

Bab ini menguraikan langkah-langkah yang dilakukan dalam penyelesaian penelitian ini.

**BAB IV : Hasil dan Pembahasan**

Pada bab ini membahas tentang hasil penelitian.

**BAB V : Penutup**

Pada bab ini menguraikan tentang kesimpulan dari keseluruhan laporan tugas akhir dan saran untuk peningkatan kualitas penelitian kedepannya.

* 1. **Waktu penelitian**

# BAB IV HASIL DAN PEMBAHASAN

1. **Hasil Penelitian**

Penelitian ini menggunakan data primer yang diperoleh secara langsung dari beberapa toko kosmetik di Palu. Dataset yang diperoleh terdiri dari 140 gambar kemasan produk *skincare* dengan berbagai jenis produk seperti pembersih wajah, toner, serum, *moisturizer*, *sunscreen*, *day cream* dan *eye cream*. Contoh dataset dapat dilihat pada Gambar 4.1, sementara dataset lengkap tersedia di Lampiran 1.

|  |
| --- |
|  |

* 1. **Preprocessing Gambar**

Pada tahap preprocessing gambar, beberapa metode diterapkan untuk meningkatkan kualitas gambar sebelum dilakukan ekstraksi teks. Tahapan *preprocessing* meliputi:

1. **Konversi *grayscale***

Konversi gambar ke grayscale bertujuan untuk menghilangkan informasi warna dan hanya mempertahankan intensitas cahaya dalam gambar. Proses ini dilakukan dengan menggabungkan nilai intensitas dari tiga channel warna utama, yaitu merah (*Red*), hijau (*Green*), dan biru (*Blue*), menggunakan rumus tertentu.

Sebagai contoh, diberikan sebuah matriks citra berwarna berukuran 32×32 dengan pemisahan tiap channel warna sebagai berikut:

|  |  |
| --- | --- |
| Matriks *Red* (R) | = |
| Matriks *Green* (G) | = |
| Matriks *Blue* (B) | = |

Perhitungan nilai grayscale dilakukan secara berurutan berdasarkan ukuran matriks. Dimulai dari elemen pertama (baris ke-1, kolom ke-1), kemudian dilanjutkan ke elemen berikutnya dalam satu baris hingga seluruh baris selesai. Setelah itu, proses berlanjut ke baris berikutnya hingga seluruh elemen dalam matriks dikonversi ke grayscale.

Sebagai contoh, perhitungan untuk elemen pertama f(1,1)f(1,1)f(1,1) dilakukan sebagai berikut:

|  |  |
| --- | --- |
|  | = (0,299 x 191) + (0,587 x 182) + (0,114 x 167)  = 57,109 + 106,834 + 19,038  = 182,981 |

Sehingga, matriks hasil *grayscale* yang diperoleh:

|  |  |
| --- | --- |
| *Grayscale* | = |

Namun, dalam penelitian ini, proses konversi dilakukan secara otomatis menggunakan bahasa *Python*. Salah satu hasil dari proses ini ditunjukkan pada Gambar 4.1.1, sedangkan data lengkap hasil konversi *grayscale* dapat dilihat pada Lampiran 2.

|  |  |
| --- | --- |
|  |  |

1. *Gaussian blur*

Gaussian Blur digunakan untuk mengurangi *noise* lebih lanjut sehingga dapat meningkatkan akurasi deteksi teks oleh OCR. Dalam penelitian ini, proses *Gaussian Blur* diterapkan secara otomatis menggunakan bahasa *Python*.

Namun, untuk memberikan pemahaman lebih lanjut mengenai cara kerja proses ini, berikut disajikan contoh perhitungan manual menggunakan rumus *Gaussian Blur* dengan kernel berukuran 3×3.

Untuk kernel 3×3, koordinat titik-titiknya didefinisikan sebagai berikut:

Selanjutnya, nilai *Gaussian* dihitung untuk setiap titik dengan parameter standar deviasi σ=1\sigma=1. Sebagai contoh, perhitungan untuk titik tengah (0,0) dilakukan sebagai berikut:

|  |  |
| --- | --- |
| *G*(0,0) | = |
|  | = = (1) = 0.1592 |

Setelah proses ini diulang untuk semua titik dalam kernel, diperoleh matriks kernel Gaussian yang belum dinormalisasi sebagai berikut:

*G* =

Selanjutnya, menghitung total bobot dari semua elemen dalam kernel:

|  |  |
| --- | --- |
| Total bobot | = 0.0585 + 0.0965 + 0.0585 + 0.0965 + 0.1592 + 0.0965 |
|  | + 0.0585 + 0.0965 + 0.0585 |
|  | = 0.07792 1 |

Karena total bobot belum bernilai 1, normalisasi diperlukan dengan membagi setiap elemen dalam kernel dengan jumlah totalnya. Normalisasi ini bertujuan untuk memastikan bahwa hasil konvolusi tidak mengubah tingkat kecerahan keseluruhan gambar, sehingga gambar yang telah difilter tetap mempertahankan tampilan aslinya dan tetap mengikuti distribusi *Gaussian*.

|  |  |
| --- | --- |
| *G*normalisasi (-1,-1), (1,-1), (-1,1), (1,1) | = 0.0750 |
| *G*normalisasi (0,-1), (-1,0), (1,0), (0,1) | = 0.1238 |
| *G*normalisasi (0,0) | = 0.2043 |

Setelah normalisasi, kernel Gaussian dapat disederhanakan ke dalam bentuk pecahan agar lebih mudah digunakan dalam pemrosesan citra.

*G* =

Setelah memperoleh kernel Gaussian, langkah selanjutnya adalah menerapkannya pada citra *grayscale*. Dalam penelitian ini, digunakan matriks intensitas piksel *grayscale* berukuran 32×32. Namun, untuk mempermudah pemahaman, hanya sebagian kecil matriks yang ditampilkan, yaitu pojok kiri atas (3×3), guna memberikan ilustrasi mengenai proses perhitungan *Gaussian Blur*.

Matriks intensitas *grayscale* awal (3×3 dari 32×32):

Setiap piksel dalam citra baru dihitung dengan mengalikan setiap elemen dalam kernel dengan elemen yang sesuai dalam matriks citra asli, kemudian menjumlahkan hasil perkaliannya. Sebagai contoh, perhitungan untuk piksel (2,2) dilakukan sebagai berikut:

|  |  |
| --- | --- |
| *I’*(2,2) | = (182,981 x 1) + (183,981 x 2) + (180,622 x 1) + (182,981 x 2) + (183,981 x 4) + (164,622 x 2) + (181,981 x 1) + (177,981 x 2) + (180,91 x 1) |
|  | =182,981 + 367,962 + 180,622 + 365,962 + 735,924 + 329,244 + 181,981 + 355,962 + 180,91 |
|  | = = 180,0968 |

Setelah semua perhitungan pada matriks dilakukan, setiap piksel dalam citra telah diproses menggunakan Gaussian Blur. Berikut adalah hasil akhir dari *Gaussian Blur* yang diterapkan pada seluruh matriks, dengan potongan bagian pojok kiri atas (3×3) sebagai ilustrasi:

***I*’ =**

Salah satu hasil *Gaussian Blur* menggunakan bahasa *python* ditunjukkan pada Gambar 4.3, sedangkan hasil lengkapnya dapat dilihat pada Lampiran 3.

|  |  |
| --- | --- |
|  |  |

1. *Thresholding*

Setelah proses Gaussian Blur diterapkan untuk menghaluskan citra, langkah selanjutnya adalah melakukan thresholding pada hasil citra tersebut. Nilai ambang (*threshold*) diterapkan untuk memisahkan teks dari latar belakang. Dalam penelitian ini, digunakan metode *Binary Otsu Thresholding,* yang secara otomatis menentukan nilai ambang optimal berdasarkan histogram citra. Proses thresholding ini dilakukan menggunakan bahasa *Python*. Contoh hasil *thresholding* ditunjukkan pada Gambar 4.3, sementara data lengkapnya dapat dilihat pada Lampiran 4.

|  |  |
| --- | --- |
|  |  |

* 1. **Segmentasi Gambar**

Tahap segmentasi dilakukan untuk membagi teks menjadi bagian yang lebih kecil agar dapat dikenali oleh sistem OCR dengan lebih baik.

1. Segmentasi baris

Dilakukan untuk memisahkan setiap baris teks dalam gambar. Proses ini dilakukan dengan mendeteksi perubahan intensitas piksel secara horizontal. Hasil segmentasi baris dapat dilihat pada gambar 4.3.1

**Gambar segmentasi baris**

1. Segmentasi kata

Setelah segmentasi baris, dilakukan segmentasi kata untuk memisahkan kata-kata dalam setiap baris menggunakan deteksi jarak antar karakter. Hasil segmentasi kata dapat dilihat pada gambar 4.3.2

**Gambar segmentasi kata**

* 1. **Ektraksi teks**

Setelah *preprocessing* selesai, dilakukan ekstraksi teks menggunakan Tesseract OCR.

|  |
| --- |
|  |

* 1. **Hasil Akhir Sistem**

**TAMPILAN WEBSITE**

* 1. **Pengujian Sistem**

Pengujian dilakukan dengan membandingkan teks hasil ekstraksi OCR dengan teks referensi yang diperoleh secara manual dari kemasan produk. Setelah dilakukan pengujian pada 140 dataset gambar kemasan produk *skincare*, diperoleh hasil sebagai berikut:

**Tabel hasil akurasi**

# BAB V KESIMPULAN DAN SARAN