BAB II
LANDASAN TEORI

2.1 TINJAUAN PUSTAKA

Sebagai usaha untuk menguatkan topik penelitian, penulis melakukan analisis dari hasil riset penelitian terdahulu yang berkaitan dengan topik penelitian. Berikut adalah hasil dari penelitian tersebut :

a. Wahyuddin, Askar Hakim (2023) dengan judul penelitian "Aplikasi Ekstraksi Data Kartu Vaksin Berbasis *Web* Menggunakan Metode OCR". Penelitian tersebut membahas tentang penggunaan metode OCR sebagai metode rekognisi dari gambar yang diambil lewat kamera untuk menghasilkan aplikasi ekstraksi data kartu vaksin dan dimunculkan dalam bentuk *website*. Berdasarkan kesimpulan penelitian tersebut hasil ekstraksi kartu vaksin menggunakan OCR berhasil dengan catatan gambar yang diunggah memiliki pencahayaan yang baik dan jelas(Wahyuddin *and* Hakim 2023).
b. Syahri Muharom (2019) dengan judul penelitian "Pengenalan Nomor Ruangan Menggunakan Kamera Berbasis OCR Dan *Template Matching*". Penelitian tersebut membahas tentang pengaplikasian metode OCR untuk dapat mengenali nomor ruangan dan menjadi alternatif metode baru untuk dapat mengenali nomor ruangan menggunakan kamera. Berdasarkan kesimpulan penelitian tersebut memiliki Tingkat akurasi yang cukup tinggi sebesar 93,75% (Muharom 2019).
c. Susan Siti Nurhaliza, Lussiana ETP (2022) dengan judul penelitian "Sistem Pengenalan Karakter Dokumen Secara Otomatis Menggunakan Metode *Optical Character Recognition*". Penelitian tersebut membahas tentang pengaplikasian metode OCR untuk mengenali huruf pada dokumen distribusi izin alat Kesehatan. Berdasarkan kesimpulan penelitian tersebut penerapan metode OCR berhasil mengenali karakter sebanyak 312 sehingga Tingkat akurasi pengujian adalah 98.78% dengan rata-rata waktu proses untuk mengenali karakter sebesar 1.29 detik (Nurhaliza and ETP 2022).
d. Muhammad Rizal Toha, Agung Triayudi (2022) dengan judul "Penerapan Membaca Tulisan di dalam Gambar Menggunakan Metode OCR Berbasis *Website* pada e-KTP". Penelitian tersebut membahas tentang penerapan metode OCR untuk membangun sebuah sistem informasi pembaca teks dari gambar foto e-KTP dan melakukan pengujian dan analisis pada hasil sistem informasi yang dibangun.

Berdasarkan kesimpulan penelitian tersebut bahwa perangkat lunak yang dirancang berhasil untuk mengekstrak semua data pada E-KTP dengan nilai akurasi yang cukup bagus (Rizal Toha *and* Triayudi 2022).

e. Hasan Nindya Murwato, Suci Aulia, Atik Novianti (2020) dengan judul penelitian "Perancangan *Translator Image to Text* Dengan Menggunakan Metode *Optical Character Recognition* Berbasis *Matlab*". Penelitian tersebut membahas tentang penerapan metode *Optical Character Recognition* untuk sistem *translator* yang dapat menerjemahkan kata dalam gambar lalu menampilkannya dengan kata yang bisa dimengerti oleh wisatawan. Berdasarkan kesimpulan penelitian tersebut bahwa metode OCR berhasil untuk melakukan translasi dengan nilai akurasi rata-rata yang cukup tinggi dengan Cahaya yang memadai karena hal tersebut sangat mempengaruhi proses deteksi karakter (Nindya Murwato, Aulia, *and* Novianti 2020).
f. Yunimawar Niati Gulo (2022) dengan judul penelitian "Penerapan Algoritma *Hamming Distance* Untuk Pencarian Teks Pada Aplikasi Ensiklopedia Indonesia". Penelitian tersebut membahas tentang penerapan metode *Hamming distance* untuk membantu pencarian teks sehingga memudahkan pencarian teks yang sesuai dengan konteks yang dicari. Berdasarkan kesimpulan penelitian tersebut pencarian teks menggunakan metode *Hamming distance* bisa diterapkan sesuai dengan penelitian yang dilakukan (Gulo 2022).
g. Mayesti Anggelina, Lucia Dwi Krisnawati, Danny Sebastian (2022) dengan judul penelitian "Penerapan *Simhash* dan *Hamming distance* dalam Deteksi kemiripan Teks Berita". Penelitian tersebut membahas tentang cara penerapan simhash dengan algoritma *Local Sensitive Hashing* juga metode *Hamming Distance* untuk membuat sistem yang bisa menampilkan dokumen yang memiliki kesamaan pada bagian nama hingga paragraf dengan Tingkat kemiripan tertentu. Berdasarkan kesimpulan penelitian tersebut Bahwa penerapan algoritma *simhash* dan *Hamming Distance* berhasil dengan Tingkat akurasi rata-rata 27% dan *recall* 80%(Anggelina, Dwi Krisnawati, and Sebastian 2022).
h. Mudawil Qulub, Rifqi Hammad, Pahrul Irfan, Yuliana (2023) dengan judul penelitian "*Improvement of Spelling Correction Accuracy in Indonesian Language through the Application of Hamming Distance Method*". Penelitian tersebut membahas tentang penerapan dan Analisa metode *Hamming Distance* terhadap kesalahan kata baku dan tidak baku dalam Bahasa Indonesia untuk mengetahui

Tingkat koreksi pada kesalahan kata bahasa Indonesia. Berdasarkan Kesimpulan penelitian tersebut pada pengujian 1 dengan kata salah atau perbedaan 1 dan 2 karakter, metode *Hamming distance* menghasilkan akurasi sebesar 98,33%. Untuk perbedaan 1 karakter 100% akurasi dan perbedaan 2 karakter 96.67% (Qulub, Hammad, *and* Irfan 2023).

i. Aria Novitra (2023) dengan judul penelitian "Penerapan Algoritma *Approximate String Matching* Untuk Pencarian Teks Pada Aplikasi Ensiklopedia Teknologi Komputer". Penelitian tersebut membahas tentang cara penerapan metode *Approximate String Matching* dan *Hamming distance* ke sebuah sistem untuk membantu pengguna sistem mencari kesalahan dalam teks. Berdasarkan kesimpulan penelitian tersebut bahwa metode *Hamming Distance* dapat diterapkan di sistem tersebut (Novitra 2023).
j. Susi Rianti, Riza Adrianti Supono (2023) dengan judul penelitian "Perbandingan Algoritma *Edit Distance*, Levenshtein Distance, *Hamming Distance, Jaccard Similarity* Dalam Mendeteksi *String Matching*". Penelitian tersebut membahas tentang cara membanding kan beberapa algoritma untuk menentukan mana yang terbaik. Dengan *Levenshtein Distance* di tempat pertama dengan nilai MAP 13,125ms dan *Hamming Distance* dengan nilai MAP 14,25ms (Rianti *and* Supono n.d.).

Berdasarkan penelitian yang telah dilakukan sebelumnya, memperkuat dimungkinkannya menemukan atau memberikan alternatif kata yang paling sesuai dengan citra tulisan tangan (tulisan dalam gambar).

2.2 LANDASAN TEORI

2.2.1 TULISAN

Tulisan merupakan salah satu bentuk komunikasi yang menggunakan Simbol atau karakter tertulis untuk menyampaikan ide, memberi informasi hingga berekspresi. Menulis melibatkan penyusunan simbol tertulis seperti huruf, angka, atau tanda baca, sesuai dengan aturan dan konvensi tertentu yang dapat dipahami oleh pembaca.

Tulisan punya peran penting dalam menyimpan, menyampaikan, dan memperluas pengetahuan manusia. Contoh tulisan itu ada di mana saja, mulai dari catatan harian, surat, esai, naskah sastra, laporan ilmiah, dan banyak lagi. Proses

menulis melibatkan pemilihan kata, struktur kalimat, dan organisasi ide untuk mencapai komunikasi yang efektif.

Selain itu, tulisan dapat menjadi bentuk seni dan ekspresi diri, memungkinkan penulis untuk mengungkapkan pemikiran, perasaan, dan pandangan mereka. Dengan menggunakan sistem tulisan tertentu, seperti alfabet atau aksara, masyarakat dapat mengembangkan dan mentransmisikan budaya, sejarah, dan pengetahuan dari generasi ke generasi.

Sejarah tulisan berawal dari perkembangan sistem penulisan oleh peradaban kuno. Salah satu pencapaian awal dalam sejarah tulisan adalah penggunaan prasasti batu oleh bangsa Mesopotamia pada sekitar 3500 SM, yang kemudian diikuti oleh pengembangan aksara piktografi oleh peradaban Mesir Kuno. Peradaban Mesir mengembangkan sistem hieroglif yang rumit, sedangkan di Mesopotamia, masyarakat *Sumeria* menggunakan papan tanah liat untuk menulis menggunakan aksara *kuneiform*. Sistem penulisan berbasis alfabet pertama kali muncul di Timur Tengah, dengan alfabet *Fenisia* yang menjadi dasar bagi banyak sistem tulisan di seluruh dunia, termasuk alfabet Yunani dan alfabet Latin yang digunakan dalam bahasa Inggris saat ini. Seiring waktu, evolusi teknologi membawa perkembangan media tulisan, mulai dari gulungan naskah hingga buku cetak, dan akhirnya mencapai era digital dengan kemunculan komputer dan perangkat elektronik. Sejarah tulisan mencerminkan evolusi masyarakat dan teknologi, menjadi fondasi untuk penyimpanan dan penyebaran pengetahuan.

Tulisan dalam konteks digital telah mengalami transformasi signifikan seiring dengan perkembangan teknologi informasi. Era digital memfasilitasi produksi, distribusi, dan konsumsi tulisan melalui berbagai platform elektronik seperti internet, media sosial, dan aplikasi daring. Pengguna dapat dengan cepat membuat, membagikan, dan mengakses tulisan secara global, memperluas jangkauan dan dampak informasi. Format digital juga memungkinkan adanya interaksi dan keterlibatan pengguna, dengan komentar, umpan balik, dan kolaborasi yang melibatkan pendengar secara langsung. Selain itu, alat-alat otomatisasi dan kecerdasan buatan semakin terlibat dalam pengembangan dan peningkatan kualitas tulisan digital. Hal ini menciptakan lingkungan yang dinamis dan terus berkembang di mana penulis dapat berinteraksi dengan pembaca secara instan, menciptakan keterlibatan yang lebih mendalam dalam dunia tulisan digital.

2.2.2 CITRA DIGITAL

Citra adalah representasi objek dua dimensi dari dunia visual, menyangkut berbagai macam disiplin ilmu yang mencakup seni, *human vision*, astronomi, teknik, dan sebagainya (Desmon Hutahaean, Dwi Waluyo, *and* Rais 2019). Citra dapat direpresentasikan dalam bentuk digital maupun tercetak(Nindya Murwato et al. 2020). Digital adalah suatu bentuk representasi informasi yang diwujudkan dalam bentuk data numerik atau digit.

Citra digital merupakan kumpulan angka–angka dalam dua dimensi. Angka – angka pada citra merupakan hasil kuantifikasi dari intensitas tingkat kecerahan dari masing-masing piksel penyusun citra. Piksel merupakan bagian elemen terkecil penyusun citra, jumlah piksel per unit panjang dalam citra dikenal sebagai resolusi citra. Semakin tinggi resolusi suatu citra maka jumlah piksel penyusunnya akan semakin banyak (Nindya Murwato et al. 2020).

Sejarah citra digital dimulai pada pertengahan abad ke-20, ketika Russell Kirsch menciptakan komputer pertama yang dapat menghasilkan gambar digital pada tahun 1950-an. Pada era ini, sensor *Charge-Coupled Device* (CCD) dikembangkan oleh George Smith dan Willard Boyle pada akhir 1960-an, memberikan landasan teknologi utama untuk sensor kamera digital. Pada tahun 1970-an, kamera digital pertama muncul, meskipun masih terbatas dalam penggunaan industri dan militer. Pada 1990-an, kamera digital mulai merambah ke pasar konsumen dengan harga yang lebih terjangkau. Perkembangan teknologi fotografi digital semakin pesat pada 2000-an, dipopulerkan oleh peningkatan kualitas kamera dan kemunculan ponsel pintar dengan kamera terintegrasi. Pada 2010-an, perkembangan *deep learning* dan algoritma pemrosesan citra digital memberikan dampak signifikan pada pengenalan objek dan analisis gambar. Pada tahun 2020-an, penggunaan citra digital semakin melibatkan realitas virtual dan *augmented*, menciptakan pengalaman visual yang semakin mendalam di era digital ini.

Salah satu yang berhubungan dengan citra digital adalah pemrosesan atau pengolahan citra digital.

2.2.3 PEMROSESAN CITRA DIGITAL

Pengolahan Citra digital adalah proses yang memungkinkan teknologi komputer untuk mengolah data digital dari suatu citra. Teknologi pengolahan gambar digunakan dengan sangat sering dalam bidang penelitian (Budiman, 2023).pengolahan citra digital mempunyai beberapa teknik yang bisa mengolah citra untuk diproses dan dianalisis sehingga menghasilkan data yang bisa digunakan nantinya dalam pengolahan citra digital.

Pengolahan Citra digital dapat dilakukan dengan mudah dengan banyaknya aplikasi yang ada sekarang, contohnya di komputer bisa menggunakan adobe *photoshop* hingga *coreldraw* sedangkan di *smartphone* bisa menggunakan aplikasi seperti *picsArt*. Untuk analisis citra digital kita juga bisa menggunakan beberapa aplikasi yang memang dikhususkan untuk hal tersebut seperti MATLAB. Pengolahan atau pemrosesan citra digital sering juga berhubungan dengan warna, mulai dari model warna yang sering didengar seperti RGB, RGBA, CMYK, HSV, HSI, HSL dan *Grayscale* hingga beberapa model warna yang jarang didengar seperti LAB, XYZ dan YUV.

2.2.4 CROPPING

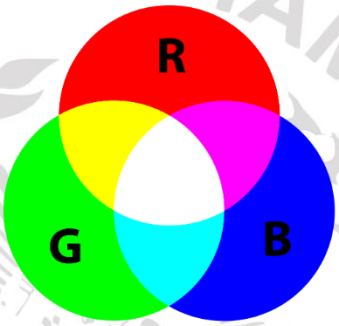
Cropping adalah proses pemotongan gambar untuk memfokuskan citra di area tertentu atau mengubah proporsinya. Ini adalah teknik yang umum digunakan dalam pengolahan citra untuk menghilangkan bagian yang tidak diinginkan atau untuk menyesuaikan komposisi gambar. Banyak alat *cropping* citra yang tersedia mulai dari *software desktop adobe* hingga *software* yang ada dalam *website* juga. Salah satu alat *cropping* yang sering dipakai dan tersedia adalah *snipping tool*.

2.2.5 RGB

RGB merupakan singkatan dari *Red*, *Green*, *and Blue* atau Merah, Biru dan hijau. Kombinasi dari warna tersebut menghasilkan bermacam-macam warna. RGB merupakan model warna dasar yang banyak digunakan dalam dunia digital karena warna tersebut berdasarkan cahaya primer. Model warna RGB digunakan untuk merepresentasikan warna pada perangkat-perangkat elektronik seperti layar komputer, televisi, kamera digital, dan berbagai perangkat tampilan lainnya. Jika dalam sebuah layer atau ruangan tersebu gelap atau hitam maka jika kita masukkan ke dalam skala RGB yaitu 0 sampai 255 maka nilai yang dihasilkan adalah

RGB(0,0,0). skala RGB disebutkan di atas memakai angka 0 sampai 255 karena hal tersebut merupakan representasi yang dapat diwakili dalam sistem biner. Sistem bilangan biner menggunakan angka 0 dan 1 dan dalam kasus RGB kita menggunakan 8-bit untuk setiap warna.

Dengan menggunakan 8-bit, nilai yang didapatkan adalah 2^8 jika dilakukan perhitungan maka mendapatkan nilai sebanyak 256 untuk setiap nilai warna. Jika nilai tersebut dimulai dari 0 kita mencapai nilai maksimum di angka 255 (0-255). Jadi jika kita menginginkan warna gambar menjadi merah, maka nilai RGB yang dimasukkan adalah RGB(255,0,0), jika warna tersebut hanya hijau maka RGB(0,255,0) hingga jika warna gambar hanya biru saja maka RGB(0,0,255), untuk warna putih maka nilai RGB yang dimasukkan adalah RGB(255,255,255).



Gambar 2.1 Warna *RGB*

2.2.6 GRAYSCALE

Grayscale adalah mode warna dalam dunia pengolahan citra di mana setiap piksel dalam citra direpresentasikan hanya oleh satu saluran warna, yaitu intensitas ke abuan atau kecerahan. Dalam mode *grayscale*, kanal warna tidak diperhitungkan, dan setiap piksel hanya memiliki tingkat ke abuan tunggal yang mencerminkan tingkat intensitas cahaya pada posisi tersebut.

Dalam citra *grayscale*, intensitas ke abuan biasanya diukur dalam skala nilai, di mana nilai minimum mewakili warna hitam (0) dan nilai maksimum mewakili warna putih (biasanya 255 dalam skala 8-bit). Nilai-nilai di antara keduanya mencerminkan tingkat kecerahan yang beragam. Mode *grayscale* sering digunakan ketika warna tidak diperlukan untuk analisis atau ketika perlu mengurangi kompleksitas data dalam citra. Ini juga umum digunakan dalam situasi di mana informasi warna tidak diperlukan dan hanya tingkat kecerahan atau intensitas yang relevan.

Proses konversi dari citra warna seperti citra RGB ke citra *grayscale*, seperti yang telah dijelaskan sebelumnya, mengambil rata-rata dari intensitas warna merah, hijau, dan biru untuk setiap piksel guna menghasilkan intensitas ke abuan tunggal dengan persamaan:

$$GS_{(x,y)} = 0.2989 * R_{(x,y)} + 0.5870 * G_{(x,y)} + 0.1141 * B_{(x,y)} \quad (2.1)$$

Atau persamaan dengan pembulatan 3 angka di belakang titik atau koma:

$$GS_{(x,y)} = 0.298 * R_{(x,y)} + 0.587 * G_{(x,y)} + 0.114 * B_{(x,y)} \quad (2.2)$$

2.2.7 ENHANCEMENT

Enhancement dalam citra merupakan serangkaian teknik yang digunakan untuk meningkatkan kualitas atau daya lihat suatu citra. Tujuan dari *enhancement* adalah membuat citra lebih jelas, meningkatkan detail, dan meningkatkan informasi visual yang dapat diambil dari citra tersebut. Beberapa teknik yang sering digunakan adalah *Sharpening, smoothing* atau *blurring, filtering, histogram enhancement* dan lainnya. Penggunaan *enhancement* harus dilakukan secara hati-hati dikarenakan jika melakukan *enhancement* secara berlebihan dapat membuat gambar menjadi terdistorsi atau peningkatan *noise* yang tidak diinginkan sehingga merusak gambar.

2.2.8 OPTICAL CHARACTER RECOGNITION

Optical Character Recognition, pengenalan karakter atau yang bisa disebut OCR adalah salah satu bidang dalam *computer vision* dan hubungan komputer manusia. OCR bekerja dengan cara memisahkan kalimat menjadi kata dan simbol huruf sebelum dilakukan pengenalan. Karena kualitas pengenalan sangat bergantung terhadap kualitas gambar dokumen tersebut maka dilakukan beberapa *pre-processing* untuk membuat algoritma dapat menangkap deteksi lebih baik seperti *Pre-processing* (Kumar Siliwangi *and* Prabowo 2022).

Sejarah *Optical Character Recognition* (OCR) dimulai pada awal abad ke-20 dengan konsep-konsep awal, seperti prototipe "*Optophone*" yang dikembangkan pada tahun 1920 untuk membantu tunanetra membaca teks. Perkembangan komersial terjadi pada dekade 1960-an ketika perusahaan-perusahaan seperti IBM dan RCA memperkenalkan sistem OCR pertama yang

dapat membaca teks dari dokumen cetak, khususnya untuk pemrosesan cek dan formulir. Selama dekade 1980-an dan 1990-an, kemajuan dalam pemrosesan gambar dan komputer meningkatkan kinerja OCR, memungkinkannya mengenali berbagai jenis huruf dan *font*. Dalam era digital pada tahun 2000-an dan seterusnya, OCR semakin terintegrasi dalam berbagai aplikasi dengan kemampuan pengenalan bahasa dan tulisan tangan. Teknologi ini terus berkembang, memanfaatkan kecerdasan buatan dan jaringan saraf untuk meningkatkan akurasi dalam mengonversi teks dari format cetak ke format digital.

OCR bisa digunakan di mana saja mulai dari pengarsipan dokumen, pengolahan formulir, pencarian teks, pengolahan tagihan dan faktur, pengolahan surat elektronik, pengenalan tulisan tangan dan lainnya. OCR juga sudah terintegrasi dengan beberapa Bahasa pemrograman, membuat OCR menjadi lebih mudah dalam penerapannya ke aplikasi yang ingin dibuat. Bahasa pemrograman yang terintegrasi antara lain *python*, *java*, C, C++, dan lainnya serta *MATLAB*.

2.2.9 MATLAB

2.2.9.1 Asal MATLAB

MATLAB adalah program komputer interaktif yang berfungsi sebagai "laboratorium" yang nyaman untuk melakukan perhitungan yang melibatkan matriks. Program ini memberikan akses mudah ke perangkat lunak matriks yang dikembangkan oleh proyek LINPACK dan EISPACK. Kemampuannya mencakup tugas-tugas standar seperti menyelesaikan persamaan linear simultan dan menginversi matriks, hingga permasalahan nilai *eigen* simetris dan non simetris, serta alat matriks yang cukup canggih seperti dekomposisi nilai *singular*.

Penggunaan utama MATLAB diharapkan terdapat di lingkungan kelas. Program ini dapat bermanfaat dalam kursus pengantar aljabar linear terapan, serta kursus lanjutan dalam analisis numerik, teori matriks, statistika, dan aplikasi matriks pada disiplin lainnya. Di luar lingkungan akademis, MATLAB dapat berperan sebagai "kalkulator meja" untuk solusi cepat masalah kecil yang melibatkan matriks.

Program ini ditulis dalam bahasa *Fortran* dan dirancang untuk dengan mudah di *instal* di bawah sistem operasi mana pun yang memungkinkan eksekusi interaktif program *Fortran*. Sumber daya yang dibutuhkan relatif

sederhana, dan dengan penggunaan yang tepat, program ini dapat dijalankan pada sebuah minikomputer dengan hanya 32K *byte* memori. Ukuran matriks yang dapat diatasi oleh MATLAB tergantung pada jumlah penyimpanan yang dialokasikan saat sistem dikompilasi pada mesin tertentu.

Secara fungsional, MATLAB menyerupai SPEAKEASY dan, dalam beberapa hal, APL. Semua merupakan bahasa terminal interaktif yang biasanya menerima perintah atau pernyataan satu baris, memprosesnya secara langsung, dan mencetak hasilnya. Namun, MATLAB hanya memiliki tipe data matriks (meskipun skalar, vektor, dan teks merupakan kasus khusus), sistem yang mendasarinya bersifat portabel dan membutuhkan sumber daya yang lebih sedikit, serta sub rutin pendukung yang lebih kuat dan, dalam beberapa kasus, memiliki sifat numerik yang lebih baik.

LINPACK dan EISPACK mewakili *state-of-the-art* dalam perangkat lunak untuk perhitungan matriks. EISPACK adalah paket dari lebih dari 70 sub rutin *Fortran* untuk berbagai perhitungan nilai *eigen* matriks yang sebagian besar didasarkan pada prosedur Algol yang diterbitkan oleh Wilkinson, Reinsch, dan rekan-rekan mereka. LINPACK adalah paket dari 40 sub rutin *Fortran* (dalam masing-masing dari empat tipe data) untuk menyelesaikan dan menganalisis persamaan linear simultan dan masalah matriks terkait. MATLAB tidak terlalu memperhatikan efisiensi waktu eksekusi atau penghematan penyimpanan, sehingga mengabaikan sebagian besar properti matriks khusus yang dimanfaatkan oleh sub rutin LINPACK dan EISPACK. Oleh karena itu, hanya 8 sub rutin dari LINPACK dan 5 dari EISPACK yang benar-benar terlibat.

Sepanjang kariernya, Moler sangat tertarik pada sistem persamaan linear, yang sering kali direpresentasikan dalam bentuk matriks. Setelah menyelesaikan doktornya di Universitas Stanford pada tahun 1965, Moler terus bekerja dengan pembimbing tesis doktoralnya (dan pendiri Departemen Ilmu Komputer Universitas Stanford) George Forsythe untuk meningkatkan metode komputasi dalam menyelesaikan sistem persamaan linear, dan mereka bersama-sama menulis sebuah buku.

Dasar matematis dan komputasional untuk versi pertama MATLAB dimulai dengan serangkaian makalah oleh J. H. Wilkinson dan 18 koleganya yang diterbitkan antara tahun 1965 dan 1970 dalam jurnal Numerische

Mathematik. Makalah-makalah tersebut dikumpulkan dalam sebuah volume yang diedit oleh Wilkinson dan C. Reinsch, "*Handbook for Automatic Computation*, *Volume II: Linear Algebra*," yang diterbitkan pada tahun 1971. Makalah-makalah tersebut menyajikan algoritma yang diimplementasikan dalam Algol 60 untuk menyelesaikan masalah persamaan linear matriks dan nilai *eigen*. Ini adalah makalah penelitian yang menyajikan hasil tentang stabilitas numerik, detail implementasi, dan, dalam beberapa kasus, metode baru. Pentingnya menggunakan transformasi ortogonal sejauh mungkin ditekankan oleh Wilkinson dan penulis lainnya. Bagian I dari *Handbook*, dengan 40 prosedur Algol, membahas masalah persamaan linear; bagian II, dengan 43 prosedur, membahas masalah nilai *eigen* (Moler *and* Little 2020).

2.2.9.2 EISPACK

Pada tahun 1970, bahkan sebelum *Handbook* diterbitkan, sebuah kelompok di Laboratorium Nasional Argonne mengajukan proposal kepada *National Science Foundation* (NSF) Amerika Serikat untuk "mengeksplorasi metodologi, biaya, dan sumber daya yang diperlukan untuk menghasilkan, menguji, dan menyebarkan perangkat lunak matematika berkualitas tinggi, serta menguji, mengesahkan, menyebarkan, dan mendukung paket perangkat lunak matematika dalam beberapa area masalah tertentu." Setiap musim panas selama 15 tahun, Wilkinson memberikan kuliah dalam kursus singkat di Universitas Michigan dan kemudian mengunjungi Laboratorium Nasional Argonne selama seminggu atau dua. Proyek tersebut mengembangkan EISPACK (*Matrix Eigensystem Package*) dengan menerjemahkan prosedur Algol untuk masalah nilai eigen dari bagian II dari *Handbook* ke dalam bahasa Fortran dan melakukan uji coba serta portabilitas secara intensif. Pada tahun 1971, versi pertama dari koleksi ini dikirimkan ke sekitar dua puluh universitas dan laboratorium nasional di mana individu-individu telah menunjukkan minat dalam proyek dan setuju untuk menguji perangkat lunak tersebut. Pada tahun 1976, versi kedua siap untuk didistribusikan secara publik. Selain sub rutin yang berasal dari bagian II dari *Handbook*, versi kedua juga mencakup algoritma SVD dari bagian I dan algoritma QZ baru untuk masalah nilai eigen generalisasi yang melibatkan dua matriks (Moler *and* Little 2020).

2.2.9.3 LINPACK

Pada tahun 1975, ketika EISPACK hampir selesai, Jack Dongarra, Pete Stewart, Jim Bunch, dan Cleve Moler mengajukan kepada NSF proyek penelitian lain yang menyelidiki metode pengembangan perangkat lunak matematika. Produk sampingan dari proyek ini akan menjadi perangkat lunak itu sendiri, yang dinamakan LINPACK untuk *Linear Equation Package*. Proyek ini juga berpusat di Argonne. Tiga peserta dari universitas bekerja di institusi mereka masing-masing selama tahun akademik, dan keempatnya berkumpul di Argonne pada musim panas.

Pengembangan LINPACK dimulai dalam bahasa *Fortran*; ini tidak melibatkan terjemahan dari Algol. Paket ini berisi 44 sub rutin dalam masing-masing dari empat presisi, REAL, DOUBLE, COMPLEX, dan COMPLEX*16. *Fortran* pada saat itu terkenal dengan kode yang tidak terstruktur, sulit dibaca, dan seperti "*spaghetti*". Para penulis mengadopsi gaya pemrograman yang ter disiplin dan mengharapkan orang serta mesin dapat membaca kode-kode tersebut. Ruang lingkup dari *loop* dan konstruksi *if-then-else* ditunjukkan dengan hati-hati dengan indentasi. *Go-to's* dan nomor pernyataan hanya digunakan untuk keluar dari blok dan menangani *loop* yang mungkin kosong. Semua *loop* dalam dilakukan dengan panggilan ke BLAS, *Basic Linear Algebra Subprograms*, yang dikembangkan bersama oleh Chuck Lawson dan rekan-rekannya di *Jet Propulsion Laboratory Caltech*. Pada sistem yang tidak memiliki kompilator *Fortran* yang dioptimalkan, BLAS dapat diimplementasikan dengan efisien dalam bahasa mesin. Pada mesin vektor, seperti CRAY-1, *loop* adalah satu instruksi vektor tunggal. Dua sub program paling penting adalah hasil kali dalam dua vektor, DDOT, dan vektor ditambah skalar kali vektor, DAXPY. Semua algoritma berorientasi kolom untuk sesuai dengan penyimpanan *Fortran* dan dengan demikian memberikan lokalitas akses memori.

Dalam suatu arti, proyek LINPACK dan EISPACK dianggap gagal. Peneliti telah mengajukan proyek penelitian kepada NSF untuk "mengeksplorasi metodologi, biaya, dan sumber daya yang diperlukan untuk menghasilkan, menguji, dan menyebarkan perangkat lunak matematika berkualitas tinggi." Mereka tidak pernah menulis laporan atau makalah yang

mengatasi tujuan tersebut. Mereka hanya menghasilkan perangkat lunak. Untuk ringkasan isi LINPACK, lihat. Hari ini, LINPACK lebih dikenal sebagai *benchmark* daripada sebagai perpustakaan perangkat lunak matriks (Moler *and* Little 2020).

2.2.9.4 MATLAB KLASIK

Pada tahun 1970-an dan awal 1980-an, saat bekerja pada proyek LINPACK dan EISPACK, Moler menjadi Profesor Matematika dan kemudian Ilmu Komputer di *University of New Mexico* di Albuquerque. Ia mengajar mata kuliah aljabar linear dan analisis numerik. Moler ingin agar mahasiswanya dapat dengan mudah mengakses fungsi-fungsi LINPACK dan EISPACK tanpa harus menulis program *Fortran*. Dengan "mudah diakses," ia berarti menghindari pemrosesan *batch* jarak jauh dan proses pengeditan-kompilasi-*link*-muat-eksekusi yang sering kali diperlukan di komputer pusat kampus. Ini berarti program harus menjadi interpreter interaktif, beroperasi dalam sistem *time-sharing* yang mulai tersedia.

Jadi, Moler mempelajari buku Niklaus Wirth berjudul "*Algorithms + Data Structures = Programs*" dan belajar cara mengurai bahasa pemrograman. Wirth menamakan bahasa contoh yang digunakannya PL/0. Ini adalah *subset* pedagogis dari Pascal milik Wirth, yang pada gilirannya, merupakan tanggapannya terhadap Algol. Mengutip Wirth, "Dalam ranah tipe data, PL/0 tetap mematuhi tuntutan kesederhanaan tanpa kompromi: bilangan bulat adalah satu-satunya tipe datanya."

Mengikuti pendekatan Wirth, Moler menulis MATLAB pertama - nama tersebut adalah singkatan dari *Matrix Laboratory* - dalam *Fortran*, sebagai dialek PL/0 dengan matriks sebagai satu-satunya tipe data. Proyek ini hanyalah hobi, aspek baru pemrograman yang dipelajarinya dan sesuatu yang bisa digunakan oleh mahasiswanya. Tidak ada dukungan resmi dari luar dan tentu saja tidak ada rencana bisnis. *MathWorks* masih beberapa tahun ke depan.

Hanya sekitar satu lusin sub rutin dari LINPACK dan EISPACK yang diperlukan karena semuanya adalah matriks kompleks. Rutinitas *output* tidak akan mencetak bagian imajiner dari entri jika setiap entri dalam matriks memiliki bagian imajiner nol. Sebuah vektor adalah matriks dengan satu

kolom atau baris. Sebuah skalar adalah matriks berukuran 1x1. Sebuah *string* teks hanya merupakan cara untuk menulis vektor baris dari kode karakter numerik.

MATLAB pertama ini bukanlah bahasa pemrograman; itu hanya merupakan kalkulator matriks interaktif sederhana. Tidak ada fungsi yang didefinisikan pengguna, tidak ada *toolbox*, tidak ada grafik. Dan tidak ada ODEs atau FFTs. Tangkapan layar dari layar awal yang ditunjukkan pada Gambar 4 mencantumkan semua fungsi dan kata-kata yang dipesan. Hanya ada 71 di antaranya. Jika Anda ingin menambahkan fungsi lain, Anda bisa meminta kode sumber dari Moler, menulis sub rutin *Fortran*, menambahkan nama baru ke tabel *parsing*, dan mengkompil ulang MATLAB (Moler *and* Little 2020).

2.2.10 TESSERACT ENGINE

Tesseract Engine adalah sebuah sebuah sistem pengenalan karakter optik yang bersumber terbuka. Berawal di tahun 1984 sebagai project Phd yang disponsori oleh HP, *tesseract* berkembang pesat dari tahun ke tahun. Dimulai dengan *scanner* awal dengan akurasi 99.5% tapi hanya bisa merekognisi 8 *font* dari *IBM Selectric "golf-ball" typewriters*. Hampir selama 2 dekade, pengenalan karakter optik digunakan secara umum untuk membantu memasukkan tulisan secara otomatis ke dalam sistem komputer. tapi selama ini sistem *Tesseract OCR* konvensional masih tidak bisa untuk membaca selain beberapa *font*. Untuk itu diperlukan sebuah perubahan, dari tahun 1987 hingga tahun 1994 *Tesseract* berkembang mulai dari perkembangan untuk menerima lebih banyak *fonts* hingga mempercepat proses *tesseract OCR*. Tapi pada tahun 1995 hingga tahun 2005 Tesseract tidak mengalami peningkatan dan hanya membuat *port* untuk *windows*. Tapi dengan ada *port windows* tersebut, *Tesseract* mulai berkembang lagi dengan bantuan *google* di tahun 2006 hingga menambahkan beberapa Bahasa ke dalam sistem *tesseract* hingga 6 bahasa. Perkembangan OCR yang terekam berhenti pada tahun 2016 dengan tambahan LSTM pada sistem *Tesseract*.

2.2.11 PYTHON

Python adalah bahasa pemrograman tingkat tinggi, interpretatif, dan umum yang dikembangkan oleh Guido van Rossum pada tahun 1991. Kata "*Python*" memiliki asal yang unik yaitu berasal dari sebuah film yang saat itu sang pengembang sedang membaca sebuah skrip yang berjudul *Monty's Python Flying* yang berasal dari BBC *Series* di negara inggris. Saat membacanya sang pengembang mendapat sebuah ide untuk menamai Bahasa pemrogramannya menjadi *Python* untuk nama yang pendek dan unik (Dhruv, Patel, *and* Doshi 2020).

Python dirancang dengan filosofi kesederhanaan dan keterbacaan kode, sehingga sangat cocok untuk pengembangan perangkat lunak, pemrograman web, pengolahan data, kecerdasan buatan, dan berbagai bidang lainnya. Keunggulan *Python* meliputi sintaksis yang bersih dan mudah dipahami, dukungan untuk berbagai paradigma pemrograman (prosedural, berorientasi objek, dan fungsional), serta memiliki sejumlah besar pustaka dan modul yang mendukung berbagai tugas.

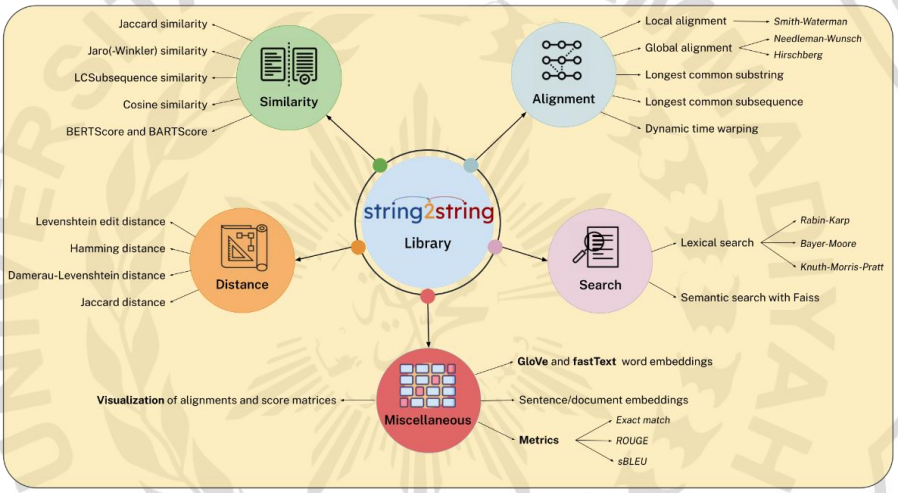
2.2.12 PERSAMAAN KATA

Persamaan kata dalam citra adalah salah satu tantangan utama dalam bidang pengolahan citra dan pengenalan pola. Persamaan kata ini terjadi ketika sebuah kata dalam citra memiliki kemiripan yang tinggi dengan kata-kata lainnya dalam aspek bentuk, struktur, dan komposisi hurufnya. Tantangan pengenalan persamaan kata ini terletak pada kompleksitas variasi dalam gaya tulisan, *font*, ukuran, dan orientasi kata, yang dapat mengakibatkan kesulitan dalam membedakan kata yang mirip.

Persamaan kata memiliki 2 konsep penting yaitu kesamaan (*Similarity*) dan Jarak (*Distance*). Dalam pengenalan kata, kesamaan digunakan untuk mengukur seberapa mirip atau serupa dua kata dalam citra berdasarkan fitur-fitur tertentu seperti bentuk, ukuran, tekstur, atau pola. Semakin tinggi nilai kesamaan antara dua kata, semakin mirip atau identik kedua kata tersebut. metode kesamaan seperti *Cosine Similarity* atau *Jaccard Similarity* sering digunakan dalam pengenalan kata untuk membandingkan kemiripan antara kata-kata dalam citra.

Di sisi lain, Jarak dalam pengenalan kata digunakan untuk mengukur jarak atau perbedaan antara dua kata dalam citra. Metode Jarak seperti *Levenshtein Distance* atau *Damerau-Levenshtein Distance* mengukur jumlah operasi yang diperlukan untuk mengubah satu kata menjadi kata lainnya, termasuk penambahan,

penghapusan, atau penggantian karakter. Semakin kecil nilai Jarak antara dua kata, semakin mirip atau serupa kata-kata tersebut dalam citra. Metode Jarak ini berguna dalam menghitung kesalahan atau perbedaan antara kata yang dikenali dengan kata referensi dalam proses pengenalan kata. Kedua konsep ini sangat penting dalam pengenalan kata dalam citra, di mana kesamaan membantu dalam membandingkan kemiripan visual antara kata-kata, sementara jarak digunakan untuk mengukur perbedaan atau kesalahan dalam pengenalan kata. Dengan menggunakan kombinasi kesamaan dan jarak, sistem pengenalan kata dalam citra dapat memberikan hasil yang akurat dan andal dalam mengenali dan memahami teks pada gambar atau citra digital. Konsep-konsep ini memainkan peran kunci dalam berbagai aplikasi seperti pengolahan dokumen, klasifikasi citra, atau sistem pencarian teks dalam citra.



Gambar 2.2 Jenis-jenis metode dalam persamaan kata

Menurut **Gambar 2.2** Terdapat beberapa metode yang umum digunakan dalam pengenalan dan perbandingan persamaan kata dalam citra, di antaranya adalah *Hamming Distance, Levenshtein Distance, Jaccard Distance,* dan *Damerau-Levenshtein Distance*. Setiap metode memiliki karakteristik dan aplikasi yang berbeda-beda dalam mengevaluasi kemiripan antara dua kata.

Metode pertama, *Hamming Distance*. Digunakan untuk mengukur jumlah posisi di mana dua kata berbeda dalam representasi biner mereka. *Hamming Distance* sesuai untuk kata yang memiliki panjang yang sama, dan hanya menghitung perbedaan pada posisi yang sama dalam dua kata. Metode ini sederhana namun efektif dalam mengukur kemiripan untuk kata yang telah

diwakilkan dalam bentuk biner, seperti dalam pengenalan pola atau pemrosesan citra biner.

Levenshtein Distance, juga dikenal sebagai *Edit Distance*, mengukur jumlah operasi (penambahan, penghapusan, atau penggantian karakter) yang diperlukan untuk mengubah satu kata menjadi kata lainnya. Metode ini lebih fleksibel karena dapat menangani kata dengan panjang yang berbeda dan memperhitungkan pengubahan karakter yang terjadi. *Levenshtein Distance* sering digunakan dalam aplikasi pengenalan teks yang memperhitungkan perubahan atau kesalahan dalam kata.

Jaccard Distance, di sisi lain, mengukur kemiripan antara dua himpunan atau sekumpulan elemen, dalam hal ini karakter-karakter yang terdapat dalam kata. *Jaccard Distance* dihitung sebagai jumlah elemen yang sama dibagi dengan total elemen yang ada di dua himpunan. Metode ini sering digunakan dalam perbandingan string yang tidak terstruktur dan tidak memperhatikan urutan karakter, hanya fokus pada keberadaan atau ketidakhadiran karakter dalam kata.

Terakhir, *Damerau-Levenshtein Distance* adalah variasi dari *Levenshtein Distance* yang juga memperhitungkan operasi transposisi karakter (penukaran posisi dua karakter berdekatan). Metode ini berguna dalam kasus di mana kesalahan dapat terjadi karena kesalahan ketik atau perubahan posisi karakter yang tidak berdekatan. Dengan memperhitungkan operasi transposisi, *Damerau-Levenshtein Distance* dapat memberikan hasil yang lebih baik dalam mengukur kemiripan antara dua kata.

2.2.13 HAMMING DISTANCE

Hamming distance adalah salah satu algoritma mengukur kedekatan item. Jika nilai jarak makin kecil, maka kedua item itu semakin dekat dan berlaku sebaliknya. Yang biasanya dibandingkan adalah kata dan bilangan biner (Sari, Saptono, *and* Suryani 2019).

Sejarah *Hamming Distance* dimulai pada tahun 1950 ketika matematikawan dan insinyur elektronika Richard Hamming memperkenalkan konsep ini sebagai metode pengukuran perbedaan antara dua rangkaian bit atau kata biner dengan panjang yang sama. Pengembangan *Hamming Distance* awalnya terjadi dalam konteks kode kesalahan untuk sistem telekomunikasi, di mana itu digunakan untuk mendeteksi dan memperbaiki kesalahan transmisi dalam bit-bit data. Seiring

waktu, konsep ini berkembang dan menemukan penerapan dalam berbagai bidang, termasuk pemrosesan citra, pengenalan pola, dan ilmu komputer secara umum. *Hamming Distance* menjadi alat yang penting dalam membandingkan dan mengukur kesamaan antara dua data yang direpresentasikan dalam bentuk biner, dan keberlanjutan penggunaannya mencerminkan relevansinya dalam evolusi teknologi dan aplikasi di era modern.

Hamming Distance bisa diterapkan di mana saja, salah satunya persamaan *String* atau kata. untuk persamaan tersebut kita memiliki basis algoritma dalam bentuk sebagai berikut:

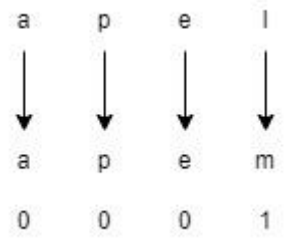
1. Cek Panjang kata dengan cara membedakan huruf antar kata.

   Kata 1 = {(a), (p), (e), (l)}

   Kata 2 = {(a), (p), (e), (m)}

   - Kata 1 punya 4 huruf dengan huruf: a, p, e dan l
   - Kata 2 punya 4 huruf dengan huruf: a, p, e dan m
2. Pilih pembanding antar kata misal jika ada 2 kata, pilih 1 kata yang digunakan sebagai pembanding kata.
3. Cek huruf dalam kata ke 2 dengan kata ke 1, jika setiap huruf dalam kata tersebut memiliki huruf yang sama maka nilainya 0 tetapi jika terjadi perbedaan huruf maka nilainya 1.

Perhitungan *hamming distance* ditunjukkan pada di bawah ini:



Gambar 2.3 Perbandingan Huruf

Berdasarkan **Gambar 2.3** dapat disimpulkan bahwa jika huruf yang berbeda dalam perbandingan kata 1 dan kata 2 ada 1 sehingga saat melakukan perhitungan yaitu semua nilai ditambahkan maka akan bernilai 1 atau dikatakan ada 1 jarak atau *distance.*