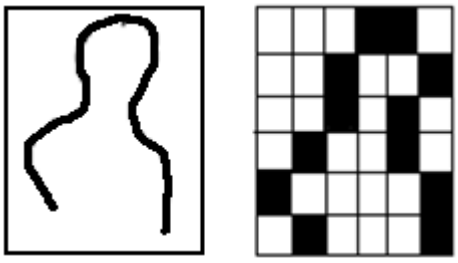
BAB 2
LANDASAN TEORI

2.1 Citra Digital

Citra digital didefinisikan sebagai fungsi f(x,y) dua dimensi, dimana x dan y adalah koordinat spasial dan f(x,y) adalah disebut dengan intensitas atau tingkat keabuan citra pada koordinat x dan y[3]. Jika x, y, dan nilai f terbatas dalam diskrit, maka disebut dengan citra digital. Citra digital dibentuk dari sejumlah elemen terbatas, yang masing-masing elemen tersebut memiliki nilai dan koordinat tertentu. *Pixel* adalah elemen citra yang memiliki nilai yang menunjukkan intensitas warna[8].

Citra digital dapat dibagi menjadi dua jenis. Jenis pertama adalah citra digital yang dibentuk oleh kumpulan *pixel* dalam array dua dimensi. Citra jenis ini disebut citra bitmap (*bitmap image*) atau citra raster (*raster image*). Jenis citra yang kedua adalah citra yang dibentuk oleh fungsi-fungsi geometri dan matematika. Jenis citra ini disebut grafik vektor (*vector graphics*).



Gambar 2.1. Citra analog dan citra digital.

2.2 Pengolahan Citra Digital

Pengolahan citra merupakan proses pengolahan dan analisis citra yang banyak melibatkan persepsi visual. Proses ini mempunyai ciri data masukan dan informasi keluaran yang berbentuk citra.

Istilah pengolahan citra digital secara umum didefinisikan sebagai pemrosesan citra dua dimensi dengan komputer. Citra digital adalah barisan bilangan nyata maupun kompleks yang diwakili oleh bit-bit tertentu.

Umumnya citra digital berbentuk persegi panjang atau bujur sangkar dengan lebar dan tinggi tertentu. Ukuran ini biasanya dinyatakan dalam banyaknya titik atau *pixel* sehingga ukuran citra selalu bernilai bulat. Setiap *pixel* memiliki koordinat sesuai posisinya dalam citra. Koordinat ini biasanya dinyatakan dalam bilangan bulat positif, yang dapat dimulai dari 0 atau 1 tergantung pada sistem yang digunakan. Setiap titik juga memiliki nilai berupa angka digital yang merepresentasikan informasi yang diwakili oleh titik tersebut.

Beberapa pemrosesan gambar yang bisa dilakukan seperti memperbaiki kualitas gambar, dilihat dari aspek radiometrik (peningkatan kontras, transformasi warna, restorasi citra) dan dari aspek geometrik (rotasi, translasi, skala, transformasi geometrik), melakukan pemilihan ciri citra (*feature images*) yang optimal untuk tujuan analisis, melakukan proses penarikan informasi atau deskripsi obyek atau pengenalan obyek yang terkandung pada citra, melakukan kompresi atau reduksi data untuk tujuan penyimpanan data, transmisi data, dan waktu proses data.

2.3 *Computer Vision*

Terminologi lain yang berkaitan erat dengan pengolahan citra adalah *computer vision* atau *machine vision*. *Computer vision* mencoba meniru cara kerja sistem visual manusia (*human vision*)[4]. *Human vision* sesungguhnya sangat kompleks. Manusia melihat objek dengan indera penglihatan (mata), lalu citra objek diteruskan ke otak untuk diinterpretasi sehingga manusia mengerti objek apa yang tampak dalam pandangan matanya. Hasil interpretasi ini mungkin digunakan untuk pengambilan keputusan.

Computer Vision merupakan proses otomatis yang mengintegrasikan sejumlah besar proses untuk persepsi visual, seperti akuisisi citra, pengolahan citra, analisis citra, pengenalan (*recognition*) dan membuat keputusan.

Proses-proses yang terlibat dalam *computer vision* adalah sebagai berikut:

1. Memperoleh atau mengakuisisi citra digital
2. Melakukan teknik komputasi untuk memproses atau memodifikasi data citra (operasi-operasi pengolahan citra)
3. Menganalisis dan menginterpretasi citra dan menggunakan hasil pemrosesan untuk tujuan tertentu.
4. *Image understanding* yang digunakan untuk mengenali pola dan membuat keputusan.

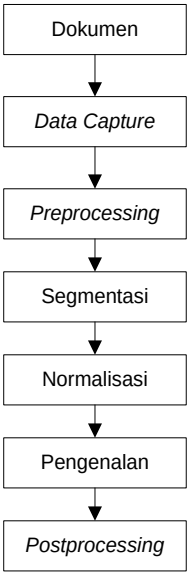
Pengolahan citra merupakan tahap *preprocessing* (proses awal) pada *computer vision*, sedangkan pengenalan pola merupakan proses untuk menginterpretasikan citra. Teknik-teknik di dalam pengenalan pola memainkan peranan penting dalam *computer vision* untuk mengenali objek.

2.4 Optical Character Recognition

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.

OCR adalah aplikasi yang menerjemahkan gambar karakter (*image character*) menjadi bentuk teks dengan cara menyesuaikan pola karakter per baris dengan pola yang telah tersimpan dalam *database* aplikasi.

Hasil dari proses OCR adalah berupa teks sesuai dengan gambar *output scanner* dimana tingkat keakuratan penerjemahan karakter tergantung dari tingkat kejelasan gambar dan metode yang digunakan[5]. Secara umum blok diagram kerja OCR dapat dilihat pada gambar 2.2.



Gambar 2.2. Blok diagram kerja OCR.

1. *Data Capture*

   Data capture merupakan proses konversi suatu dokumen (*hardcopy*) menjadi suatu *file* gambar digital.

2. *Preprocessing*

   Preprocessing merupakan suatu proses untuk menghilangkan bagian-bagian yang tidak diperlukan pada gambar input untuk proses selanjutnya.

3. Segmentasi

   Segmentasi adalah proses memisahkan area pengamatan (*region*) pada tiap karakter yang akan dideteksi.

4. Normalisasi

   Normalisasi adalah proses merubah dimensi *region* tiap karakter. Dalam OCR algoritma yang digunakan pada proses ini adalah algoritma *scaling*.

5. Pengenalan

   Pengenalan merupakan proses untuk mengenali karakter yang diamati dengan cara membandingkan ciri-ciri karakter yang diperoleh dengan ciri-ciri karakter yang ada pada *database*.

6. *Postprocessing*

   Pada umumnya proses yang dilakukan pada tahap ini adalah proses koreksi ejaan sesuai dengan bahasa yang digunakan.

2.4.1 *Image Preprocessing*

Untuk mengambil informasi dari *pixel* yang ada pada gambar, setiap komponen pada sistem pengenalan karakter dirancang untuk mengurangi jumlah data. Langkah pertama adalah *image preprocessing* yang bertujuan untuk mengubah intensitas *pixel* gambar agar mudah digunakan pada proses selanjutnya.

Tahapan yang dilakukan dalam *image preprocessing* adalah *grayscale* dan binerisasi.

2.4.1.1 *Grayscale*

Dalam komputasi, suatu citra digital *grayscale* atau *greyscale* adalah suatu citra dimana nilai dari setiap *pixel* merupakan *sample* tunggal. Citra yang ditampilkan dari citra jenis ini terdiri atas warna abu-abu, bervariasi pada warna hitam pada bagian yang intensitas terlemah dan warna putih pada intensitas terkuat. Citra *grayscale* berbeda dengan citra ”hitam-putih”, dimana pada konteks komputer, citra hitam putih hanya terdiri atas dua warna saja yaitu ”hitam” dan ”putih” saja. Citra *grayscale* disimpan dalam format delapan bit untuk setiap sample *pixel*, yang memungkinkan sebanyak 256 intensitas. Nilai intensitas paling rendah merepresentasikan warna hitam dan nilai intensitas paling tinggi merepresentasikan warna putih.

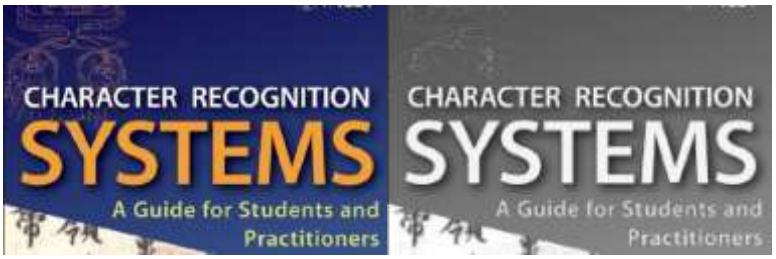
Citra *grayscale* digunakan untuk menyederhanakan model citra. Citra berwarna terdiri dari tiga *layer* matrik yaitu R- *layer*, G-*layer* dan B-*layer*. Sehingga untuk melakukan proses-proses selanjutnya tetap diperhatikan tiga *layer* tersebut. Bila setiap proses perhitungan dilakukan menggunakan tiga *layer,* berarti dilakukan tiga perhitungan yang sama. Konsep tersebut diubah dengan mengubah tiga *layer* menjadi satu *layer* matrik *grayscale* dan hasilnya adalah citra *grayscale*. Dalam citra ini tidak ada lagi warna, yang ada adalah derajat keabuan.

Untuk mengubah citra berwarna yang mempunyai nilai matrik masing-masing r, g dan b menjadi citra *grayscale* dengan nilai s, maka konversi dapat

dilakukan dengan mengambil rata-rata dari nilai r, g dan b sehingga dapat dituliskan pada persamaan 2-1.

$$s = \frac{r + g + b}{3} \quad \text{.....(2-1)}$$

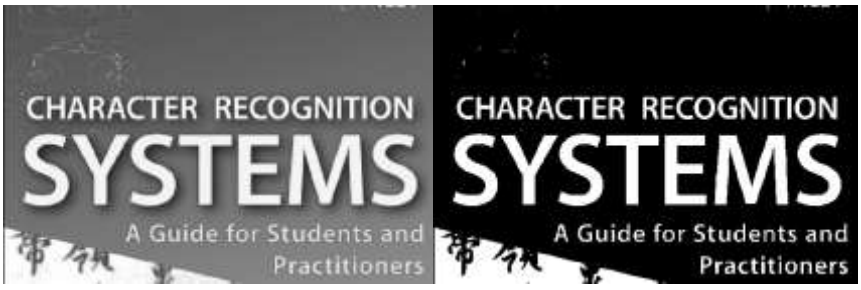
Gambar 2.3 merupakan contoh konversi citra berwarna menjadi citra grayscale dengan menggunakan persamaan 2.1.



Gambar 2.3. Hasil konversi citra berwarna menjadi citra *grayscale*.

2.4.1.2 Binerisasi

Binerisasi adalah proses mengubah citra berderajat keabuan menjadi citra biner atau hitam putih sehingga dapat diketahui daerah mana yang termasuk obyek dan *background* dari citra secara jelas. Input untuk proses *tresholding* adalah *grayscale image*. Output dari proses ini adalah *binary image.* Gambar 2.4 merupakan ilustrasi konversi citra keabuan menjadi citra biner.



Gambar 2.4. Hasil konversi citra *grayscale* menjadi citra biner.

Thresholding digunakan untuk memisahkan bagian citra yang dibutuhkan menghilangkan bagian citra yang tidak dibutuhkan [2]. Dalam pelaksanaannya *thresholding* membutuhkan suatu nilai yang digunakan sebagai nilai pembatas antara intensitas objek objek utama dengan latar belakang, dan nilai tersebut dinamakan dengan *threshold*.

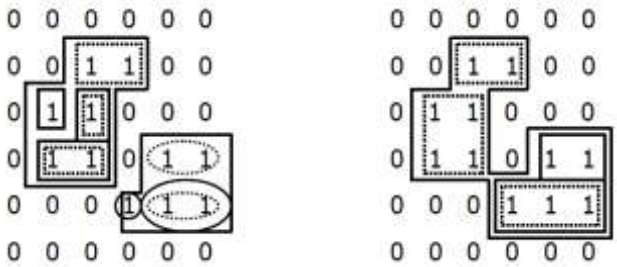
Thresholding digunakan untuk mempartisi citra dengan mengatur nilai intensitas semua piksel yang lebih besar dari nilai *threshold T* sebagai latar depan dan yang lebih kecil dari nilai *threshold T* sebagai latar belakang atau sebaliknya.

2.4.2 Segmentasi

Segmentasi merupakan suatu proses untuk mendapatkan area atau obyek yang diinginkan pada suatu citra dengan memisahkan area atau objek dari latar belakangnya. Cara yang digunakan untuk memisahkan objek yang akan dikenali adalah dengan memeriksa intensitas pixel di sekitar objek.

2.4.3 *Connected component analysis*

Proses *connected component analysis* memeriksa pixel-pixel di sekitar objek. Citra di-*scan* secara mendatar dari kiri ke kanan, dimulai dari atas citra hingga ke bawah citra untuk mencari pixel yang memiliki nilai yang sama dengan pixel di sekitarnya. Pada proses ini apabila ditemukan pixel hitam, akan dilakukan pengecekan nilai pixel hitam di sekitarnya. Jika semua pixel yang berada di sekitar pixel hitam tersebut memiliki nilai yang sama, maka pixel tersebut merupakan kesatuan dari suatu objek.



Gambar 2.5. Teknik analisis pixel tetangga.

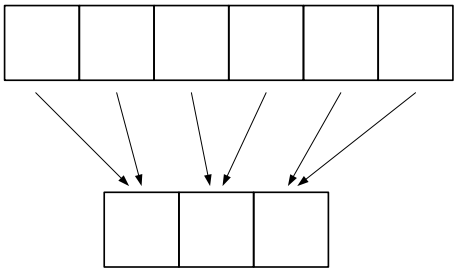
Gambar 2.5 merepresentasikan teknik analisis pixel tetangga. Analisis pixel-pixel tetangga bisa dilakukan dengan empat tetangga atau delapan tetangga. Analisis menggunakan teknik empat tetangga dilakukan dengan memeriksa pixel sebelah atas, bawah, kiri, dan kanan pixel. Dengan teknik tersebut, pixel tetangga dengan intensitas yang sama namun berada pada posisis diagonal akan dianggap objek lain. Sedangkan analisis dengan teknik delapan tetangga, pixel tetangga dengan intensitas yang sama namun berada pada posisi diagonal, akan dianggap objek yang sama. Gambar 2.5 sebelah kiri merepresentasikan analisis pixel tetangga dengan menggunakan teknik empat tetangga. Sedangkan gambar 2.5 sebelah kanan merepresentasikan analisis pixel tetangga dengan menggunakan teknik delapan tetangga.

2.4.4 Normalisasi

Citra dipetakan pada *pixel* dengan ukuran tertentu sehingga memberikan representasi dimensi yang tetap. Tujuan dari normalisasi citra adalah mengurangi resolusi citra yang berguna saat proses pengenalan citra dan juga meningkatkan akurasi pengenalan. Proses yang digunakanpada tahap normalisasi ini adalah proses penskalaan citra.

2.4.4.1 Algoritma Penskalaan

Scaling atau penskalaan pada citra disebut juga *image zooming*, yaitu proses untuk mengubah ukuran citra asli (*zoom in* / memperbesar ukuran citra asli atau *zoom out* / memperkecil ukuran citra asli). Proses perubahan ukuran resolusi citra dibutuhkan untuk menyesuaiakan resolusi citra masukan dengan resolusi citra *template*.



Gambar 2.6. Perubahan ukuran *pixel*.

Gambar 2.6 mengilustrasikan perubahan ukuran *pixel* dengan ukuran awal 6x1 pixel menjadi 3x1 pixel. Garis panah yang berasal dari *pixel* lama ke *pixel* baru mengilustrasikan pemetaan koordinat setiap *pixel* lama terhadap *pixel* baru. Persamaan 2-2 digunakan untuk menentukan faktor pengali dalam menentukan koordinat y pada *pixel* baru.

$$dy = \frac{newheight}{tinggi + 1} \quad \text{.....(2-2)}$$

Persamaan 2-3 digunakan untuk menghitung faktor pengali dalam menentukan koordinat x.

$$dx = \frac{newwidth}{lebar + 1} \quad \text{.....(2-3)}$$

Persamaan 2-4 digunakan untuk menentukan koordinat x.

$$x = i * dx \quad \text{.....(2-4)}$$

Persamaan 2-5 digunakan untuk menentukan koordinat y.

$$x = i * dx \quad \text{.....(2-5)}$$

2.4.5 Pengenalan Pola

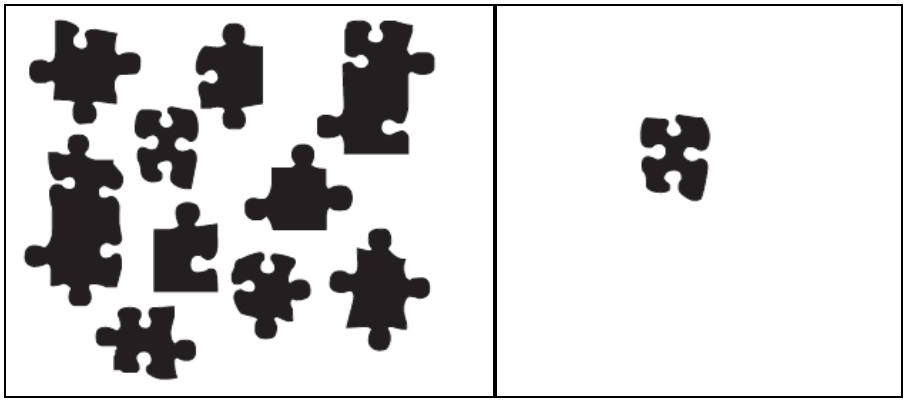
Pengenalan pola dapat dikatakan sebagai kemampuan mengenali objek berdasarkan ciri-ciri dan pengetahuan yang pernah diamatinya dari objek-objek tersebut. Tujuan dari pengenalan pola adalah mengklasifikasi dan mendeskripsikan pola atau objek kompleks melalui pengetahuan sifat-sifat atau ciri-ciri objek tersebut.

Ada tiga pendekatan dalam pengenalan pola yaitu secara sintaks, statistik, dan semantik[7]. Pengenalan pola secara sintaks dilakukan berdasarkan ciri-ciri objek. Pengenalan pola secara statistik dilakukan berdasarkan komputasi matematis. Pendekatan dengan semantik berarti pola dikenali dalam tataran yang lebih abstrak.

2.4.5.1 *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*. 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.



Gambar 2.7. Ilustrasi *template matching*.

Gambar 2.7 bagian kiri merupakan citra yang mengandung objek yang sama dengan objek pada *template* yang ada di sebelah kanan. *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 eror terkecil [3] dengan menggunakan persamaan 2-6.

$$\min e = \sum_{(x,y)\epsilon W} (I_{x,y} - T_{x,y})^2 \quad \text{.....(2-6)}$$

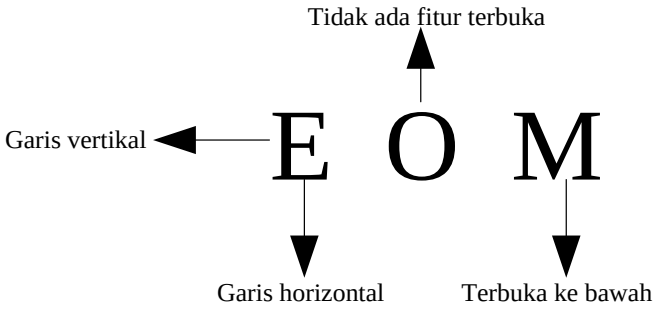
I adalah pola pixel citra masukan yang akan dibandingkan. T adala pola pixel citra *template*. *Template* dengan nilai eror paling kecil adalah *template* yang paling sesuai dengan citra masukan yang akan dibandingkan.

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* m^2 sesuai dengan semua titik citra, maka komputasi yang harus dilakukan adalah $O(N^2m^2)$. 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. Hal ini yang menyebabkan metode *template matching* menjadi lamban.

2.4.5.2 *Feature Extraction*

Feature extraction merupakan salah satu cara untuk mengenali suatu objek dengan melihat ciri-ciri khusus yang dimiliki objek tersebut[6]. Tujuan dari *feature extraction* adalah melakukan perhitungan dan perbandingan yang bisa digunakan untuk mengklasifikasikan ciri-ciri yang dimiliki oleh suatu citra.



Gambar 2.8. Ilustrasi *feature extraction.*

Gambar 2.8 merupakan ilustrasi citra karakter dengan ciri-cirinya. Ciri-ciri dari masing-masing citra *template* akan di simpan. Citra masukan yang akan dibandingkan akan dianalisis berdasarkan ciri-ciri citra. Ciri-ciri yang dimiliki citra masukan akan diklasifikasikan terhadap ciri-ciri citra *template*.

2.4.6 Klasifikasi

Jika tujuan dari *feature extraction* adalah memetakan pola berdasarkan ciri-ciri yang dimiliki oleh suatu citra, maka klasifikasi bertujuan untuk mengenali citra dengan cara mengklasifikasikan ciri-ciri yang dimilikinya.

Klasifikasi adalah proses untuk menemukan model atau fungsi yang menjelaskan atau membedakan konsep atau kelas data, dengan tujuan untuk dapat memperkirakan kelas dari suatu objek yang labelnya tidak diketahui [3].

Proses klasifikasi biasanya dibagi menjadi dua fase yaitu fase *learning* dan fase *test*. Pada fase *learning*, sebagian data yang telah diketahui kelas datanya diumpankan untuk membentuk model perkiraan. Kemudian pada fase *test* model yang sudah terbentuk diuji dengan sebagian data lainnya untuk mengetahui akurasi dari model tersebut. Bila akurasinya mencukupi model ini dapat dipakai untuk prediksi kelas data yang belum diketahui.

Teknik ini dapat memberikan klasifikasi pada data baru dengan memanipulasi data yang ada yang telah diklasifikasi dan dengan menggunakan hasilnya untuk menghitung jarak antara ciri-ciri citra *template* dan citra masukan.

2.4.6.1 *k-Nearest Neighbour*

k-nearest neighbor (k-NN atau KNN) bekerja dengan cara membandingkan data uji dan data training/*template* [3]. k-NN mencari pola data *template* yang paling mendekati dengan data uji. Dekat atau jauhnya tetangga dihitung berdasarkan jarak *Euclidean*. Dengan kata lain, kedua data dibandingkan berdasarkan atribut-atributnya.

k-NN memiliki beberapa kelebihan yaitu tangguh terhadap data *template* yang *noisy* dan efektif apabila jumlah data *template* besar. Sedangkan kelemahan dari k-NN adalah perlunya menentukan nilai parameter k (jumlah dari tetangga terdekat), pembelajaran berdasarkan jarak tidak jelas mengenai jenis jarak apa yang harus digunakan dan atribut mana yang harus digunakan untuk mendapatkan hasil yang terbaik, dan biaya komputasi cukup tinggi karena diperlukan perhitungan jarak dari tiap *query instance* pada keseluruhan *training sample.* Persamaan 2-7 adalah persamaan yang digunakan untuk menghitung jarak [3].

$$d(p,q)^2 = d(q,p)^2 = \sum_{i=1}^{n} (q_i - p_i)^2 \quad \text{.....(2-7)}$$

Nilai *k* yang terbaik untuk algoritma ini tergantung pada data. Pada umumnya, nilai *k* yang tinggi akan mengurangi efek *noise* pada klasifikasi, tetapi membuat batasan antara setiap klasifikasi menjadi lebih kabur.

Langkah-langkah yang digunakan pada metode k-NN adalah sebagai berikut:

1. Tentukan parameter k. k adalah jumlah tetangga terdekat. Nilai k yang diguankan pada penelitian ini adalah 1.
2. Hitung jarak antara ciri-ciri citra *template* dan citra masukan dengan menggunakan persamaan 2-7.
3. Urutkan jarak dari kecil ke besar.
4. Gunakan parameter k sebagai acuan dalam mengambil sejumlah data berdasarkan urutan pada nomor 3.

Ketepatan algoritma k-NN ini sangat dipengaruhi oleh fitur-fitur unik setiap citra yang akan dikenali. Riset terhadap algoritma ini sebagian besar membahas bagaimana memilih dan memberi bobot terhadap fitur, agar performa klasifikasi menjadi lebih baik.

2.5 Kompleksitas Algoritma

Algoritma yang baik adalah algoritma yang mangkus (*efficient*)[9]. Kemangkusan algoritma diukur dari waktu (*time*) eksekusi algoritma dan kebutuhan ruang (*space*) memori.

Algoritma yang mangkus ialah algoritma yang meminimumkan kebutuhan waktu dan ruang. Kebutuhan waktu dan ruang suatu algoritma bergantung pada ukuran masukan (n), yang menyatakan jumlah data yang diproses. Kemangkusan algoritma dapat digunakan untuk menilai algoritma yang baik dari sejumlah algoritma penyelesaian masalah. Besaran yang dipakai untuk menerangkan model abstrak pengukuran waktu/ruang ini adalah kompleksitas algoritma. Ada dua macam kompleksitas algoritma, yaitu kompleksitas waktu dan kompleksitas ruang.

Kompleksitas waktu, $T(n)$, diukur dari jumlah tahapan komputasi yang dibutuhkan untuk menjalankan algoritma sebagai fungsi dari ukuran masukan n.

Kompleksitas ruang, $S(n)$, diukur dari memori yang digunakan oleh struktur data yang terdapat di dalam algoritma sebagai fungsi dari ukuran masukan n.

Kompleksitas waktu dihitung berdasarkan jumlah tahapan komputasi operasi yang dilakukan dalam sebuah algoritma. Dalam prakteknya, operasi yang

dihitung hanya operasi khas yang mendasari suatu algoritma. Misalnya, operasi khas di dalam algoritma pencarian di dalam larik adalah perbandingan elemen larik. Operasi khas algoritma pengurutan adalah perbandingan dan pertukaran elemen.

Kompleksitas waktu dibedakan atas tiga macam yaitu:

1. *Worst case* yaitu kompleksitas waktu untuk kasus terburuk. *Worst case* mengindikasikan kebutuhan waktu maksimum.
2. *Best case* yaitu kompleksitas waktu untuk kasus terbaik. *Best case* mengindikasikan kebutuhan waktu mnimum.
3. *Average case* yaitu kompleksitas waktu untuk kasus rata-rata. *Average case* mengindikasikan kebutuhan waktu rata-rata.

2.6 Basis Data dan Sistem Basis Data

Menurut James Martin (1975) istilah basis data dapat dipahami sebagai suatu kumpulan data terhubung (interrelated data) yang disimpan secara bersama-sama pada suatu media, tanpa mengatap satu sama lain atau tidak perlu suatu kerangkapan data (jikapun ada, maka kerangkapan data tersebut harus seminimal mungkin dan terkontrol), data disimpan dengan cara-cara tertentu sehingga mudah untuk digunakan atau ditampilkan kembali data dapat digunakan oleh satu atau lebih program-program aplikasi secara optimal data disimpan tanpa mengalami ketergantungan dengan program yang akan menggunakannya data disimpan sedemikian rupa sehingga proses penambahan, pengambilan dan modifikasi data dapat dilakukan dengan mudah dan terkontrol. Suatu basis data mempunyai kriteria penting yang harus dipenuhi, yaitu :

1. Berorientasi pada data (data oriented) dan bukan berorientasi pada program (program oriented) yang akan menggunakannya.
2. Data dapat digunakan oleh pemakai yang berbeda-beda arau beberapa program aplikasi tanpa perlu mengubah basis data.
3. Data dalam basis data dapat berkembang dengan mudah dan baik volume maupun strukturnya.
4. Data yang ada dapat memenuhi kebutuhan sistem-sistem baru secara mudah.
5. Data dapat digunakan dengan cara yang berbeda-beda.
6. Kerangkapan data (*data redundancy)* minimal.

Sistem basis data adalah sekumpulan subsistem yang terdiri atas basis data dengan para pemakai yang menggunakan basis data secara bersama-sama, personal-personal yang merancang dan mengelola basis data, untuk merancang dan mengelola basis data serta sistem komputer pendukungnya. Sistem basis data mempunyai beberapa elemen penting, yaitu :

1. Basis data sebagai inti dari sistem basis data.
2. Perangkat lunak (*software*) untuk perancangan dan pengelolaan basis data.
3. Perangkat keras (*hardware*) sebagai pendukung operasi pengelolaan data.
4. Manusia (*brainware*) sebagai perangkat atau para spesialis informasi yang mempunya fngsi sebagai perancang/pengelola.

2.7 Pemodelan Analisis

Model analisis merupakan representasi teknis yang pertama dari sistem, tetapi saat ini ada yang mendominasi landasan pemodelan analisis. Pertama, analisis terstruktur adalah metode pemodelan klasik, dan analisis berorientasi objek.

Analisis terstruktur adalah aktivitas pembangunan model. Analisis tersturktur menggunakan notasi yang sesuai dengan prinsip analisis operasional dapat menciptakan model yang menggambarkan muatan dan aliran informasi, membagi sistem secara fungsional dan secara *behavioral*, dan menggambarkan esensi dari apa yang harus dibangun.

Entity-relationship Diagram adalah notasi yang digunakan untuk melakukan aktivitas pemodelan data. Atribut dari masing-masing objek data yang dituluis pada ERD dapat digambarkan dengan menggunakan deskripsi objek data, sedangkan *data flow diagram (DFD)* memberikan informasi tambahan yang digunakan selama analisis domain informasi dan berfungsi sebagai dasar bagi pemodelan fungsi.

2.8 *Entity-relationship Diagram* (ERD)

ERD adalah model konseptual yang mendeskripsikan hubungan antara penyimpanan (dalam DFD)**.** ERD digunakan untuk memodelkan struktur data dan hubungan antar data. Dengan ERD, model dapat diuji dengan mengabaikan proses yang dilakukan. ERD pertama kali dideskripsikan oleh Peter Chen yang dibuat sebagai bagian dari perangkat lunak Case.

2.9 Diagram Konteks

Diagram konteks menggambarkan hubungan antara sistem dengan entitas luarnya. Diagram konteks berfungsi sebagai transformasi dari satu proses yang melakukan transformasi data *input* menjadi data data *output*.

2.10 *Data Flow Diagram* (DFD)

DFD adalah suatu model logika data atau proses yang dibuat untuk menggambarkan dari mana asal data dan kemana tujuan data yang keluar dari sistem, dimana data disimpan, proses apa yang menghasilkan data tersebut dan interkasi antara data yang tersimpan dan proses yang dikenakan pada data tersebut.

DFD sering digunakan untuk menggambarkan suatu sistem yang telah ada atau sistem baru yang akan dikembangkan secara logika tanpa mempertimbangkan lingkungan fisik dimana data tersebut mengalir atau dimana data tersebut akan disimpan.

DFD merupakan alat yang digunakan pada metodologi pengembangan sistem yang terstruktur. Kelebihan utama pendekatan alir data, yaitu :

1. Kejelasan dari menjalankan implementasi teknis sistem.
2. Pemahaman lebih jauh mengenai keterkaitan satu sama lain dalam sistem dan subsistem.
3. Mengkomunikasikan pengetahuan sistem yang ada dengan pengguna melalui diagram alir data.
4. Menganalisis sistem yang diajukan untuk menentukan apakah data-data dan proses yang diperlukan sudah ditetapkan.

DFD terdiri dari *context diagram* dan diagram rinci (DFD *leveled*), *context diagram* berfungsi memetakan model lingkungan (menggambarkan hubungan antara entitas luar, masukan dan keluaran sistem), yang direpresentasikan dengan lingkaran tunggal yang mewakili keseluruhan sistem. DFD *leveled* menggambarkan sistem jaringan kerja antara fungsi yang berhubungan satu sama lain dengan aliran data penyimpanan data, model ini hanya memodelkan sistem dari sudut pandang fungsi.

2.11 Delphi 7.0

Bahasa pemrograman Delphi merupakan pemrograman visual (berbasis windows) yang dibuat oleh sebuah Perusahaan Software Borland .Inc. Bahasa pemrograman Delphi merupakan bahasa pemrograman yang dikembangkan dari bahasa pemrograman Pascal. Fungsi dari aplikasi ini sama dengan fungsi aplikasi visual lainnya.

Delphi 7.0 dapat menangani pembuatan aplikasi sederhana hingga aplikasi berbasis jaringan. Delphi 7.0 dapat dimanfaatkan untuk membuat aplikasi berbasis teks, grafis, angka, database maupun web.

Bahasa pemrograman visual memiliki dua hal yaitu object dan kode program. Objek berbentuk komponen yang dapat dilihat (visual), sedangkan kode program merupakan sekumpulan teks yang digunakan sebagai perintah yang telah diatur dengan suatu aturan.

2.12 MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (bahasa Inggris: *database management system*) atau DBMS yang *multithread*, *multi-user*, dengan sekitar 6 juta instalasi di seluruh dunia. MySQL AB membuat MySQL tersedia sebagai perangkat lunak gratis dibawah lisensi GNU General Public License (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL.

Berbeda dengan proyek-proyek seperti Apache, dimana perangkat lunak dikembangkan oleh komunitas umum, dan hak cipta untuk kode sumber dimiliki oleh penulisnya masing-masing, MySQL dimiliki dan disponsori oleh sebuah perusahaan komersial Swedia MySQL AB, dimana memegang hak cipta hampir atas semua kode sumbernya. Kedua orang Swedia dan satu orang Finlandia yang mendirikan MySQL AB adalah: David Axmark, Allan Larsson, dan Michael "Monty" Widenius.