BAB 2
TINJAUAN PUSTAKA

2.1 Penyakit Gigi

Penyakit gigi adalah gangguan terhadap organ gigi. Penyebab penyakit gigi bervariasi mulai dari infeksi pada gigi atau gusi, retak gigi, makan yang masih tersisa pada gigi, abeses pada gigi, dan pecahnya gigi yang berjalan seiring waktu [6]. Adapun gejala penyakit gigi adalah bengkak disekitar gigi, terasa nyeri pada gigi, dan demam atau sakit kepala [6]. Akibatnya, organ gigi menjadi terganggu, bahkan jika tidak diawasi dengan baik maka dapat menyebabkan kehilangan gigi. Contoh penyakit gigi gingivitis dan periodontitis.

2.1.1 Gingivitis

Gingivitis atau radang gusi (gingiva) adalah penyakit gigi yang disebabkan akibat menumpuknya bakteri atau plak. Gingivitis adalah gejala jenis non-destruktif dari penyakit periodontal sehingga jika tidak ditangani dengan baik maka dapat berkembang menjadi penyakit periodontitis yang dapat menyebabkan hilangnya gigi. Seorang yang terkena mengalami gusi berdarah dan membengkak. Penyakit gingivitis termasuk penyakit ringan namun tetap harus segara diatasi [7]. Contoh penyakit gingivitis pada gambar 2.1.



Gambar 2.1 Penyakit Gingivitis

Gejala penyakit radang gusi adalah

1. Gusi lunak dan terkadang ngilu ketika disentuh
2. Gusi mudah berdarah
3. Bau mulut
4. Gusi bengkak (peradangan)

5. Gusi surut

Penyebab utama gusi radang atau gingivitis adalah akumulasi plak dan bakteri yang berada di sekitar gigi yang memicu respon imun, dan pada gilirannya dapat menyebabkan kerusakan jaringan gingiva, komplikasi hingga pada akhirnya bisa mengakibatkan kehilangan gigi [7]. Penyebab lainnya adalah perubahan hormon, merokok, riwayat keluarga, dan obat [7].

2.1.2 Periodontitis

Penyakit periodontitis adalah infeksi gusi serius yang merusak jaringan lunak dan tulang yang menyokong gigi[8]. Periodontitis dapat menyebabkan gigi rontok atau meningkatkan resiko penyakit stroke. Contoh penyakit periodontitis pada gambar 2.2.



Gambar 2.2 Penyakit Periodontitis

Gejala penyakit penyakit periodontitis adalah

1. Pembengkakan gusi
2. Rongga terbentuk diantara gigi
3. Gigi lepas
4. Perubahan gigi saat mengigit

Penyebab penyakit periodontitis adalah penumpukan plak atau bakteri gigi pada gigi akibat tidak menyikat gigi. Penyebab lainnya adalah karang gigi plak yang mengeras pada gigi sehingga sulit dihilangkan dan dapat mengiritasi gusi[8].

2.2 Kecerdasan Buatan

Kecerdasan Buatan atau *artificial intelligence* merupakan salah satu bagian ilmu komputer yang membuat agar mesin (komputer) dapat melakukan pekerjaan seperti dan sebaik yang dilakukan manusia. Pada awal diciptakannya, komputer hanya difungsikan sebagai alat hitung saja. Namun seiring dengan perkembangan jaman, maka peran komputer semakin mendominasi kehidupan umat manusia. Komputer tidak lagi hanya digunakan sebagai alat hitung, lebih dari itu, komputer diharapkan untuk dapat diberdayakan untuk mengerjakan segala sesuatu yang bisa dikerjakan oleh manusia.

Kecerdasan buatan digunakan untuk menganalisis pemandangan dalam citra dengan perhitungan simbol-simbol yang mewakili isi pemandangan tersebut setelah citra diolah untuk memperoleh ciri khas. Kecerdasan buatan bisa dilihat sebagai tiga kesatuan yang terpadu yaitu persepsi, pengertian dan aksi. Persepsi menerjemahkan sinyal dari dunia nyata dalam citra menjadi simbol-simbol yang lebih sederhana, pengertian memanipulasi simbol-simbol tadi untuk memudahkan penggalian suatu informasi tertentu, dan aksi menerjemahkan simbol-simbol yang telah dimanipulasi menjadi sinyal lain yang dapat merupakan hasil akhir atau hasil antara sesuai dengan keperluan [9].

2.3 Pola

Pola adalah suatu entitas yang terdefinisi dan dapat diidentifikasi serta diberi nama melalui ciri-cirinya (*feature*). Ciri-ciri tersebut digunakan untuk membedakan suatu pola dengan pola yang lainnya. Ciri yang baik adalah ciri yang memiliki daya pembeda yang tinggi, sehingga pengelompokan pola berdasarkan ciri yang dimiliki dapat dilakukan dengan keakuratan yang tinggi. Pola adalah komposit/gabungan dari ciri yang merupakan sifat dari sebuah objek [10]. Berikut dibawah ini adalah contoh pola

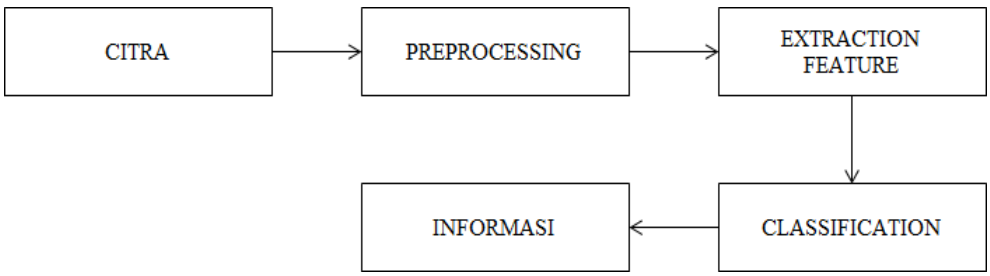
1. Huruf, memiliki ciri-ciri seperti tinggi, tebal, titik sudut, dan lengkungan garis.
2. Suara, memiliki ciri-ciri seperti amplitudo, frekuensi, nada, dan intonasi.
3. Tanda tangan, memiliki ciri-ciri seperti panjang, kerumitan, dan tekanan.
4. Sidik jari, memiliki ciri-ciri seperti lengkungan, dan jumlah garis.

Ciri pada suatu pola diperoleh dari hasil pengukuran terhadap objek uji. Khusus pada pola yang terdapat di dalam citra, ciri-ciri yang dapat diperoleh berasal dari informasi

1. Spasial seperti intensitas pixel dan histogram.
2. Tepi seperti arah dan kekuatan.
3. Kontur seperti garis, elips dan lingkaran.
4. Wilayah/bentuk seperti keliling, luas dan pusat massa.
5. Hasil transformasi Fourier seperti frekuensi.

2.4 Pengenalan Pola

Pengenalan pola merupakan suatu ilmu untuk mengklasifikasikan atau menggambarkan sesuatu berdasarkan pengukuran kuantitatif ciri atau sifat dari objek. Pola dapat berupa kumpulan hasil pengukuran yang bisa dinyatakan dalam notasi vektor atau matriks. Pola tersebut merupakan suatu entitas yang terdefinisi dan dapat di-identifikasi dan diberi nama. Struktur sistem pengenalan pola digambarkan sebagai berikut [10].



Gambar 2.3 Pengenalan Pola

Pengenalan pola mengelompokkan data numerik dan simbolik (termasuk citra) secara otomatis oleh komputer. Tujuan pengelompokan ini adalah untuk mengenali suatu objek dalam citra. Manusia bisa mengenali objek yang dilihatnya karena otak manusia telah belajar mengklasifikasikan objek-objek di alam sehingga mampu membedakan suatu objek dengan objek lainya. Kemampuan visual manusia inilah yang dicoba ditiru oleh mesin atau dalam hal ini adalah komputer. Komputer menerima input berupa citra objek yang akan diidentifikasi,

memproses citra tersebut dan memberikan output berupa deskripsi objek di dalam citra[10].

2.5 Pengolahan Citra

Pengolahan citra adalah pemrosesan citra, khususnya dengan menggunakan komputer, menjadi citra yang kualitasnya lebih baik. Umumnya, operasi-operasi pada pengolahan citra diterapkan pada citra bila

1. Perbaikan atau memodifikasi citra perlu dilakukan untuk meningkatkan kualitas penampakan atau untuk menonjolkan beberapa aspek informasi yang terkandung di dalam citra.
2. Elemen di dalam citra perlu dikelompokkan, dicocokkan, atau diukur.
3. Sebagian citra perlu digabung dengan bagian citra yang lain.

Pengolahan Citra bertujuan memperbaiki kualitas citra agar mudah diinterpretasi oleh manusia atau mesin (dalam hal ini komputer). Teknik-teknik pengolahan citra mentransformasikan citra menjadi citra lain. Jadi, masukannya adalah citra dan keluarannya juga citra, namun citra keluaran mempunyai kualitas lebih baik dari pada citra masukan [10]

2.6 Operasi Pengolahan Citra

Operasi-operasi yang dilakukan di dalam pengolahan citra banyak ragamnya. Namun, secara umum, operasi pengolahan citra dapat diklasifikasikan dalam beberapa jenis sebagai berikut [10]

1. Perbaikan kualitas citra (*image enhancement*)

Jenis operasi ini bertujuan untuk memperbaiki kualitas citra dengan cara memanipulasi parameter-parameter citra. Dengan operasi ini, ciri-ciri khusus yang terdapat di dalam citra lebih ditonjolkan [10]. Contoh-contoh operasi perbaikan citra

a. Perbaikan kontras gelap/terang
b. Perbaikan tepian objek (*edge enhancement*)
c. Penajaman (sharpening)
d. Pemberian warna semu (*pseudocoloring*)
e. Penapisan derau (noise *filtering*)

2. Pemugaran citra (image *restoration*)

Operasi ini bertujuan menghilangkan/meminimumkan cacat pada citra. Tujuan pemugaran citra hampir sama dengan operasi perbaikan citra [10]. Bedanya, pada pemugaran citra penyebab degradasi gambar diketahui. Contoh-contoh operasi pemugaran citra

a. Penghilangan kesamaran (*deblurring*)
b. Penghilangan derau (*noise*)

3. Pemampatan citra (*image compression*)

Jenis operasi ini dilakukan agar citra dapat direpresentasikan dalam bentuk yang lebih kompak sehingga memerlukan memori yang lebih sedikit. Hal penting yang harus diperhatikan dalam pemampatan adalah citra yang telah dimampatkan harus tetap mempunyai kualitas gambar yang bagus [10].

4. Segmentasi citra (*image segmentation*)

Jenis operasi ini bertujuan untuk memecah suatu citra ke dalam beberapa segmen dengan suatu kriteria tertentu. Jenis operasi ini berkaitan erat dengan pengenalan pola [10].

5. Pengorakan citra (*image analysis*)

Jenis operasi ini bertujuan menghitung besaran kuantitif dari citra untuk menghasilkan deskripsinya. Teknik pengorakan citra mengekstraksi ciri-ciri tertentu yang membantu dalam identifikasi objek. Proses segmentasi kadangkala diperlukan untuk melokalisasi objek yang diinginkan dari sekelilingnya. Contoh-contoh operasi pengorakan citra [10]

a. Pendeteksian tepi objek (*edge detection*)
b. Ekstraksi batas (*boundary*)
c. Representasi daerah (*region*)

6. Rekonstruksi citra (*image reconstruction*)

Jenis operasi ini bertujuan untuk membentuk ulang objek dari beberapa citra hasil proyeksi. Operasi rekonstruksi citra banyak digunakan dalam bidang medis. Misalnya beberapa foto rontgen dengan sinar X digunakan untuk membentuk ulang gambar organ tubuh [10].

7. Perubahan Model Warna

Warna adalah persepsi yang dirasakan oleh sistem visual manusia terhadap panjang gelombang cahaya yang dipantulkan oleh objek. Setiap warna mempunyai panjang gelombang yang berbeda. Warna merah mempunyai panjang gelombang paling tinggi, sedangkan warna ungu mempunyai panjang gelombang paling rendah. Warna-warna yang diterima oleh mata merupakan hasil kombinasi cahaya dengan panjang gelombang berbeda. Penelitian memperlihatkan bahwa kombinasi warna yang memberikan rentang warna yang paling lebar adalah red (R), green (G), blue (B). Dalam pembentukan citra digital model warna yang umum digunakan antara lain sebagai berikut

a. Citra RGB

Citra RGB yang biasa disebut juga citra true color, disimpan dalam citra berukuran (m x n) x 3 yang mendefinisikan warna merah (red), hijau (green) dan biru (blue) untuk setiap pikselnya. Warna pada setiap piksel ditentukan berdasarkan kombinasi dari warna red, green dan blue (RGB). RGB merupakan citra 24bit dengan komponen merah, hijau, dan biru yang masing-masing umumnya bernilai 8bit sehingga intensitas kecerahan warna sampai 256 level dan kombinasi warnanya kurang dari sekitar 16 juta warna [10].

b. Citra Keabuan

Citra dengan derajat keabuan berbeda dengan citra RGB, citra ini didefinisikan oleh satu nilai derajat warna. Umumnya bernilai 8 bit sehingga intensitas kecerahan warna sampai 256 level dan kombinasi warnanya 256 varian. Tingkat kecerahan paling rendah yaitu 0 untuk warna hitam dan putih bernilai 255 [10]. Ada beberapa macam cara untuk mengkonversi sistem warna RGB menjadi *grayscale*, yaitu [11]

1. Menggunakan rata-rata setiap komponen warna RGB. *Grayscale* = (R+G+B) / 3.
2. Menggunakan nilai maksimal komponen RGB. *Grayscale* = Max{R,G,B}.
3. Menggunakan sistem YUV (sistem warna pada NTSC), dengan cara mengambil komponen Y(iluminasi).

NTSC (National Television System Committee) mendefinisikan bobot untuk konversi citra *true color* ke *grayscale* sebagai berikut

wR = 0.299, wB = 0.587, wG = 0.114.

Data masukkan berupa citra *True Color* dan data keluaran berupa citra *Grayscale* [11]. Komponen Y sendiri diperoleh dari sistem warna RGB dengan konversi. Adapun persamaan nilai citra keabuan pada komponen Y adalah sebagai berikut [11]

$$Grayscale = (R\ x\ 0.299) + (\text{B X } 0.587\text{s}) + (\text{G X } 0.114) \quad (2.1)$$

Dimana

Grayscale = nilai keabuan yang diperoleh dari

R = nilai *Red* (merah)

G = nilai *green* (hijau)

B = nilai *blue* (biru)

8. Perataan Histogram (Histogram Equalization)

Menurut Rinaldi [10] Teknik perataan histogram merupakan perpaduan antara penggeseran dan pelebaran histogram. Tujuan dari perataan histogram adalah untuk mendapatkan citra dengan tingkat keabuan yang utuh dan distribusi piksel terbagi secara merata pada setiap tingkat keabuan.

Karena histogram menyatakan peluang pixel dengan derajat keabuan tertentu, maka rumus menghitung histogram ditulis kembali sebagai fungsi peluang.

$$P_r(r_k) = \frac{n_k}{n} \quad (2.2)$$

Dimana

Pr = nilai peluang munculnya piksel

nk = nilai jumlah piksel citra yang muncul pada intensitas tertentu.

n = total keseluruhan piksel.

yang dalam hal ini,

$$r_k = \frac{k}{L-1} \quad , \quad 0 \le k \le L-1 \quad (2.3)$$

Dimana

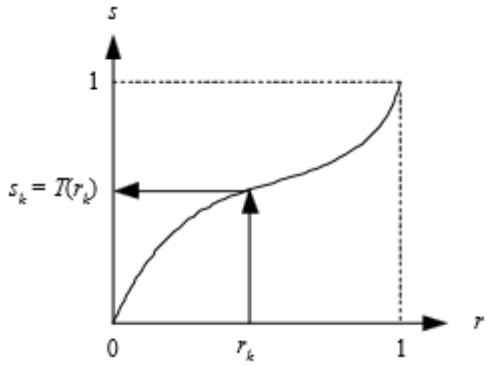
rk = nilai intensitas keabuan

k = nilai keabuan

L = derajat keabuan.

Nilai rk = 0 menyatakan hitam, dan rk = 1 menyatakan putih dalam skala keabuan yang didefinisikan. Yang dimaksud dengan perataan histogram adalah mengubah derajat keabuan suatu pixel (*r*) dengan derajat keabuan yang baru (*s*) dengan suatu fungsi transformasi T, yang dalam hal ini *s = T(r)*. [10]

Gambar 2.4 memperlihatkan transformasi r menjadi s. Tiga sifat yang dipertahankan pada transformasi ini



Gambar 2.4 Fungsi Transformasi

1. Nilai *s* merupakan pemetaan 1 ke 1 dari *r*. Ini untuk menjamin representasi intensitas yang tetap. Ini berarti *r* dapat diperoleh kembali dari *r* dengan transformasi invers.

$$\mathrm{r} = \mathrm{T}^{-1}(\mathrm{s}) \quad ,0 \ \leq 1 \tag{2.4}$$

2. Untuk $0 \leq ri \leq 1$, maka $0 \leq T(r) \leq 1$. Ini untuk menjamin pemetaan T konsisten pada rentang nilai yang diperbolehkan.
3. Semakin tinggi nilai s, tingkat keabuan semakin meningkat.

   Untuk fungsi histogram yang menerus,

$$s = T(r) = \int P_r(w)dw \quad ,0 \ \leq r \ \leq 1 \tag{2.5}$$

   yang dalam hal ini w adalah peubah bantu.

Dalam bentuk diskrit, nilai-nilai s diperoleh dengan persamaan pada nomor (2.6)

$$S_k = T(r_k) = \sum_{j=0}^{k} \frac{n_j}{n} = \sum_{j=0}^{k} P_r(r_j) \tag{2.6}$$

yang dalam hal ini, $0 \leq \mathrm{rk} \leq 1$, k = 0, 1, 2, …, L – 1

2.7 Tekstur

Secara umum tekstur mengacu pada repetisi elemen-elemen tekstur dasar yang sering disebut primitif atau texel (*texture element*). Suatu texel terdiri dari beberapa pixel dengan aturan posisi bersifat periodik, kuasiperiodik, atau acak. Pengertian dari tekstur dalam hal ini adalah keteraturan pola-pola tertentu yang terbentuk dari susunan piksel-piksel dalam citra digital [9].

Untuk membentuk suatu tekstur setidaknya ada dua persyaratan yang harus dipenuhi antara lain

1. Terdiri dari satu atau lebih piksel yang membentuk pola-pola primitif (bagian-bagian terkecil). Bentuk-bentuk pola primitif ini dapat berupa titik, garis lurus, garis lengkung, luasan dan lain-lain yang merupakan elemen dasar dari sebuah bentuk.
2. Munculnya pola-pola primitif yang berulang-ulang dengan interval jarak dan arah tertentu sehingga dapat diprediksi atau ditemukan karakteristik perulangannya.

Suatu citra memberikan interpretasi tekstur yang berbeda apabila dilihat dengan jarak dan sudut yang berbeda, manusia memandang tekstur berdasarkan deskripsi yang bersifat acak, seperti halus, kasar, teratur, tidak teratur, dan lain sebgainya. Hal ini merupakan deskripsi yang tidak tepat dan non-kuantitatif, sehingga diperlukan adanya suatu deskripsi yang kuantitatif (matematis) untuk memudahkan analisis.

2.8 Analisis Tekstur

Analisis tekstur merupakan dasar dari berbagai macam aplikasi, aplikasi dari analisis tekstur antara lain adalah penginderaan jarak jauh, pencitraan medis, identifikasi kualitas suatu bahan (kayu, kulit, tekstil dan lain-lain). Pada analisis citra, dikategorikan menjadi lima kategori utama yaitu statistis, struktural, geometri, model dasar dan pengolahan sinyal. Pendekatan statistis mempertimbangakan bahwa internsitas dibangkitkan oleh medan acak dua dimensi, metode ini berdasar pada frekuensi-frekuensi ruang. Contoh metode statistis adalah fungsi autokorelasi, run-length, matriks kookurensi, tranformasi

fourier, frekuensi tepi. Teknik struktural berkaitan dengan penyusupan bagian-bagian terkecil suatu citra.contoh metode struktural adalah model fraktal. Metode geometri berdasar atas perangkat geometri yang ada pada elemen tekstur. Contoh model dasar adalah medan acak. Sedangkan metode pengolahan sinyal adalah metode yang berdasarkan analisis frekuensi seperti transformasi gabor dan transformasi wavelet [9].

2.9 Konvolusi

Konvolusi (*convolution*) adalah sebuah proses dimana citra dimanipulasi dengan menggunakan eksternal mask atau subwindows untuk menghasilkan citra yang baru. Sedangkan *filtering* tanpa menggunakan eksternal mask tetapi hanya menggunakan piksel tetangga untuk mendapatkan piksel baru. Konvolusi banyak digunakan dalam berbagai pengolahan citra untuk *smoothing, crispening, edge detection*, serta efek lainnya.

Proses konvolusi dapat dijelaskan sebagai berikut. Kernel (*sliding window*) diletakkan pada setiap piksel dari citra input dan menghasilkan piksel baru. Nilai piksel baru dihitung dengan mengalikan setiap nilai piksel tetangga dengan bobot yang berhubungan pada *kernel* dan kemudian menjumlah hasil dari perkalian tersebut.

Untuk pengolahan citra, operasi yang dilakukan adalah operasi diskrit karena nilai koordinat piksel merupakan nilai yang diskrit. Selanjutnya *filter* atau mask yang digunakan dalam pengolahan citra biasanya berukuran terbatas, dalam arti bahwa bobot atau pengaruh dari titik-titik yang cukup jauh sudah tidak signifikan sehingga dapat diabaikan (dianggap nol) [10].

2.10 Kernel

Kernel adalah matrik yang pada umumnya berukuran kecil dengan elemen-elemennya adalah berupa bilangan. Kernel digunakan pada proses konvolusi. Oleh karena itu, kernel juga disebut dengan *convolution window* (jendela konvolusi). Ukuran kernel dapat berbeda-beda, seperti 2 x 2, 3 x 3, 5 x 5, dan sebagainya. Elemen–elemen kernel yang juga disebut bobot (weight) merupakan bilangan-bilangan yang membentuk pola-pola tertentu. Kernel biasa

juga disebut dengan tapis (*filter*), *template*, *mask*, serta *sliding window*. Dalam konsep morphologi kernel disebut juga dengan *structuring element* [10].

2.11 Filter Gabor

Ekstrasi ciri adalah pengambilan ciri sebuah objek yang dapat menggambarkan dari objek tersebut. Tujuan ekstrasi ciri adalah mencari daerah fitur yang signifikan pada citra tergantung pada karakteristik intrinsik dan aplikasi. *Filter* gabor adalah satu salah teknik ekstrasi ciri yang sering dipakai dalam pengenalan ciri. *Filter* gabor adalah fungsi *Gaussian* yang dikalikan dengan fungsi harmonic [12]. Metode *filter* gabor dibentuk dua elemen yaitu kompleks, sinusoidal (dengan lengkungan) serta Gaussian. Metode *filter gabor* menghubungkan representasi yang optimal antara arah orientasi dengan frekuensi (domain spasial). Metode *filter* gabor sering digunakan dalam pendeteksian suara, iris mata, sidik jari, analisis tekstur karena efektif dalam mengenali pola. Metode *filter gabor* dikenal sebagai detektor ciri yang sukses, memiliki kemampuan menghilangkan variabilitas wajah. Keuntungan lain menggunakan *filter gabor* adalah mampu mempresentasikan citra kedalam orientasi sudut dan frekuensi, sehingga ekstrasi citra yang dihasilkan akan lebih detil. Untuk mendapatkan Kernel *Gabor* maka digunakan persamaan berikut ini [13]

$$g_e(x,y) = \frac{1}{2\pi\ \sigma_x\sigma_y} * \exp -\frac{1}{2}\left(\frac{x^2}{\sigma_x} + \frac{y^2}{\sigma_y}\right) * \cos(\ 2\pi\omega_{x0}x\ + 2\pi\omega_{y0}y) \qquad (2.7)$$

Dimana

ω_{x0} = nilai frekuensi angular dimensi x

ω_{y0} = nilai frekuensi angular dimensi y

θn = orientasi sudut terhadap garis-garis paralel fungsi gabor.

σx = standar deviasi Gausian Envelope dimensi x

σy = standar deviasi Gausian Envelope dimensi y

x = titik kordinat x

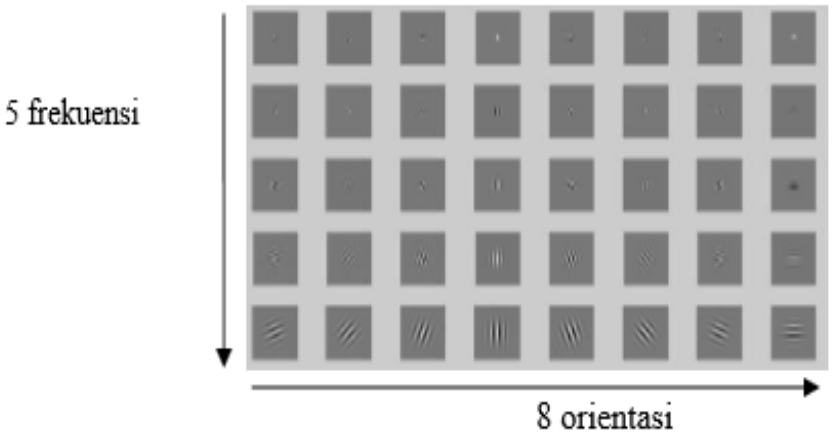
y = titik kordinat y

Dengan nilai

$x_{\theta n}$ = x cos (θn) + y sin (θn) (2.8)

$y_{\theta n}$ = - x cos (θn) + y sin (θn) (2.9)

Jika semua *gabor filter* dengan panjang gelombang (λ) dan derajat orientasi (θ) berbeda diterapkan pada satu titik tertentu (x, y), maka didapatkan banyak respon *filter* untuk titik tersebut, misal digunakan lima panjang gelombang dan delapan orientasi, maka akan dihasilkan 40 respon *filter* untuk tiap titik citra yang dikonvolusikan dengan *filter* tersebut.



Gambar 2.5 Gabor Kernel

Setelah mendapatkan ciri *gabor* maka ekstrasi ciri dapat dilakukan. Seleksi ciri memilih kuantitatif dari ciri yang ada sehingga dapat membedakan kelas-kelas objek. Ekstrasi ciri mengukur besaran kuantitatif ciri setiap piksel dengan persamaan. Salah satu contoh ekstrasi ciri yang digunakan dalam pengolahan citra adalah standar deviasi dan mean.

2.12 Algoritma K-Nearest Neighbors

Algoritma *k*-NN (*K-Nearest Neighbors)* adalah algoritma supervised learning yang melakukan klasifikasi terhadap objek berdasarkan yang jaraknya paling dekat dengan objek tersebut. Tujuan dari algoritma *k*-NN adalah mengklasifikasikan objek baru berdasarkan atribut dan training samples. Dimana hasil dari sampel uji yang baru diklasifikasikan berdasarkan mayoritas dari kategori pada *k-NN*. Proses pengklasifikasian *k*-NN didasarkan pada memori. Algoritma *k*-NN menggunakan klasifikasi ketetanggaan sebagai nilai prediksi dari sample uji yang baru. Jarak digunakan pada *k*-NN adalah jarak Euclidean yang sering digunakan pada data numeric [14].

Pada fase training, algoritma ini hanya melakukan penyimpanan vektor-vektor fitur dan klasifikasi data training sample. Pada fase klasifikasi, fitur – fitur yang sama dihitung untuk testing data (klasifikasinya belum diketahui). Jarak dari vektor yang baru ini terhadap seluruh vektor training sample dihitung, dan sejumlah k buah yang paling dekat diambil. Titik yang baru klasifikasinya diprediksikan termasuk pada klasifikasi terbanyak dari titik – titik tersebut. [14]

Nilai k yang terbaik untuk algoritma ini tergantung pada data; secara umumnya, nilai k yang tinggi akan mengurangi efek noise pada klasifikasi, tetapi membuat batasan antara setiap klasifikasi menjadi lebih kabur. Nilai k yang bagus dapat dipilih dengan optimasi parameter, misalnya dengan menggunakan cross-validation. Kasus khusus di mana klasifikasi diprediksikan berdasarkan data pembelajaran yang paling dekat (dengan kata lain, k = 1) disebut algoritma nearest neighbor. [14]

Ketepatan algoritma k-NN ini sangat dipengaruhi oleh ada atau tidaknya fitur-fitur yang tidak relevan, atau jika bobot fitur tersebut tidak setara dengan relevansinya terhadap klasifikasi. Riset terhadap algoritma ini sebagian besar membahas bagaimana memilih dan memberi bobot terhadap fitur, agar performa klasifikasi menjadi lebih baik. [14]

K buah data learning terdekat akan melakukan voting untuk menentukan label mayoritas. Label data query akan ditentukan berdasarkan label mayoritas dan jika ada lebih dari satu label mayoritas maka label data query dapat dipilih secara acak di antara label-label mayoritas yang ada. [14]

Pada proses pengklasifikasian, algoritma ini tidak menggunakan model apapun untuk dicocokkan dan hanya berdasarkan pada memori. Algoritma k-NN menggunakan klasifikasi ketetanggaan sebagai nilai prediksi dari sampel uji yang baru. Jarak yang digunakan adalah jarak Euclidean Distance. Jarak Euclidean adalah jarak yang paling umum digunakan pada data numerik. Euclidean distance didefinisikan sebagai berikut

$$d(xi, xj) = \sqrt{\sum_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2} \quad (2.10)$$

Keterangan

$d(xi, xj)$ = Jarak Euclidean

(x_i) = record ke- i

(x_2) = record ke- j

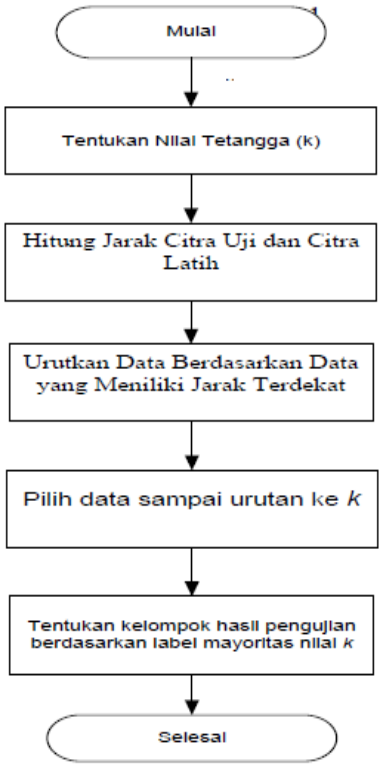
(a_r) = data ke-r

i, j = 1,2,3 … n

Adapun langkah-langkah menghitung *K-Nearest Neighbors* adalah sebagai berikut

1. Menghitung parameter K (jumlah tetangga yang paling dekat).
2. Menghitung kuadrat jarak Euclid (*query instance*) masing-masing objek terhadap data sampel yang diberikan.
3. Lalu mengurutkan objek-objek tersebut ke dalam kelompok jarak *euclidean* yang paling kecil.
4. Mengumpulkan kategori klasifikasi *K-nearest neighbors*.
5. Dengan melakukan proses perhitungan kategori klasifikasi neighbors yang paling banyak maka dapat memprediksi jarak *Euclidean*.

Algortima dari KNN untuk ditunjukan pada diagram alir berikut dibawah ini



Gambar 2.6 Flowchart Algoritma *K-Nearest Neighbors*

Keunggulan Algoritma *K-Nearest Neighbors* adalah [4]

1. Tangguh terhadap training data yang memiliki banyak *naïve bayes*
2. Efektif apabila *training* datanya besar.

Kelemahan Algoritma *k*-NN adalah

1. KNN menentukan nilai dari parameter *k* (jumlah dari tetangga terdekat).
2. *Training* berdasarkan jarak tidak jelas mengenai jenis apa yang harus
3. Atribut mana yang harus digunakan untuk mendapatkan hasil terbaik.
4. Biaya komputasi cukup tinggi karena diperlukan perhitungan jarak dari tiap *query instance* pada keseluruhan *training sample.*

2.13 Bahasa Pemrograman C#

C# (tanda '#' dibaca "Sharp") merupakan bahasa pemograman baru yang diciptakan Microsoft secara khusus sebagai salah satu bahasa pemrograman dalam teknologi .Net sebagai bahasa baru, C# tidak berevolusi dari bahasa C# versi bukan teknologi .Net. dengan demikian C# dapat memaksimalkan kemampuannya tanpa khawatir dengan masalah kompatibilitas dengan versi-versi sebelumnya.

Keharusan sebuah perangkat lunak untuk tetap dapat kompatibel dengan versi-versi sebelumnya sebagaimana yang terjadi pada Visual Basic (VB) maupun C++biasanya menghambat optimalitas kemampuan dari perangkat lunak tersebut [15].

Sejak diluncurkan pada tahun 2000, C# dengan cepat merebut hati progammer C++ bahkan VB. Dengan tata cara penulisan yang mirip C++ dan interface mirip VB 6.0 menurut wikipedia, sebuah ensiklopedia gratis di internet pengguna C# .Net pada saat ini sudah melebihi pengguna VB.Net. Sementara itu jumlah pengguna bahasa pemrograman lain masih berada dibawah jumlah pengguna VB.Net. Masih menurut wikipedia, jumlah buku C# yang terjual pun berada dikisaran 2 hingga 3 kali lebih banyak dari jumlah buku VB yang terjual.

Dari informasi ini dapat disimpulkan bahwa C# merupakan bahasa pemrogrman baru yang sedang berkembang dan dapat diterima dengan baik oleh kebanyakan progammer dan kalangan industri. Di Microsoft sendiri, C# merupakan bahasa pemrograman yang digunakan untuk membuat perangkat lunak yang berteknologi+. Net dengan demikian dapat diperkirakan bahwa C# akan menjadi bahasa pemrograman yang akan banyak digunakan di masa-masa mendatang [15].

2.14 OOP (Object Oriented Programming)

Metodologi berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang berisi data dan operasi yang diberlakukan terhadapnya. Metodologi berorientasi objek merupakan suatu cara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara sistematis. Metode berorientasi objek didasarkan pada penerapan prinsip-prinsip pengelolaan kompleksitas. Metode berorientasi objek meliputi rangkaian aktivitas analisis berorientasi objek, perancangan berorientasi objek, pemrograman berorientasi objek dan pengujian berorientasi objek [16].

Pendekatan berorientasi objek merupakan suatu teknik atau cara pendekatan dalam melihat permasalahan dan sistem (sistem perangkat lunak, sistem informasi, atau sistem lainnya). Pendekatan berorientasi objek akan memandang sistem yang akan dikembangkan sebagai suatu kumpulan objek yang berkorespondensi dengan objek-objek dunia nyata.

Sistem berorientasi objek merupakan sebuah sistem yang dibangun dengan berdasarkan metode berorientasi objek adalah sebuah sistem yang komponennya dibungkus menjadi kelompok data dan fungsi. Setiap komponen dalam sistem tersebut dapat mewarisi atribut, sifat dan komponen lainnya yang dapat berinteraksi satu sama lain [16].

Berikut ini adalah beberapa konsep dasar yang harus dipahami tentang metodologi berorientasi objek [16].

1. Kelas (*Class*)

Kelas adalah kumpulan objek-objek dengan karakteristik yang sama. Kelas merupakan definisi statik dan himpunan objek yang sama yang mungkin lahir atau diciptakan dengan kelas tersebut. Sebuah kelas akan mempunyai sifat (atribut), kelakuan (operasi/metode), hubungan (*relationship*) dan arti. Suatu kelas dapat diturunkan pada kelas yang lain, dimana atribut dan kelas semula dapat diwariskan ke kelas yang baru.

2. Objek (*Object*)

Objek adalah abstraksi dan sesuatu yang mewakili dunia nyata seperti benda, manusia, satuann organisasi, tempat, kejadian, struktur, status, atau hal-hal lain yang bersifat abstrak. Objek merupakan suatu entitas yang mampu menyimpan informasi (status) dan mempunyai operasi (kelakuan) yang dapat diterapkan atau dapat berpengaruh pada status objeknya. Objek mempunyai siklus hidup yaitu diciptakan, dimanipulasi dan dihancurkan.

3. Metode (*method*)

Metode atau method pada sebuah pada sebuah kelas hampir sama dengan fungsi atau prosedur pada metode struktural. Sebuah kelas boleh memiliki lebih dari satu metode atau operasi. Metode atau operasi yang berfungsi untuk memanipulasi objek itu sendiri. Metode merupakan fungsi atau transformasi yang

dapat dilakukan terhadap objek atau dilakukan oleh objek. Metode dapat berasal dari event, aktivitas atau aksi keadaan, fungsi, atau kelakuan dunia nyata. Contoh metode atau operasi misalnya *Read*, *Write*, *Move*, dan sebagainya. Kelas sebaiknya memiliki metode get dan set untuk setiap atribut agar konsep enkapsulasi tetap terjaga. Metode *get* digunakan untuk memberikan akses kelas lain dalam mengakses atribut, dan *set* adalah metode yang digunakan untuk mengisi atribut, agar kelas lain tidak mengakses atribut secara langsung.

4. Atribut (*attribute*)

Atribut dari sebuah kelas adalah variabel global yang dimiliki sebuah kelas. Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh objek dalam kelas objek. Atribut dimiliki secara individual oleh sebuah objek, misalnya berat, jenis, nama, dan sebagainya. Atribut sebaiknya bersifat privat untuk menjaga enkapsulasi

5. Abstraksi (*abstraction*)

Prinsip untuk merepresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.

6. Enkapsulasi (*encapsulation*)

Pembungkusan atribut data dan layanan (operasi-operasi) yang dimiliki objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerjanya.

7. Pewarisan (*Inheritance*)

Mekanisme yang memungkinkan satu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dirinya.

8. Antarmuka (*Interface*)

Antarmuka atau interface sangat mirip dengan kelas, tapi tanpa atribut kelas dan memiliki metode yang dideklarasikan tanpa isi. Deklarasi metode pada sebuah interface dapat diimplementasikan oleh kelas lain. Sebuah kelas dapat mengimplementasikan lebih dari satu antarmuka dimana kelas ini akan mendeklarasikan metode pada antarmuka yang dibutuhkan oleh kelas itu sekaligus mendefinisikan isinya pada kode program kelas itu. Metode pada antarmuka yang

diimplementasikan pada suatu kelas harus sama persis dengan yang ada pada antarmuka. Antarmuka atau interface biasanya digunakan agar kelas yang lain tidak mengakses langsung ke suatu kelas, mengakses antarmukanya.

9. Reusability

Pemanfaatan kembali objek yang sudah didefinisikan untuk suatu permasalahan pada permasalahan lainnya yang melibatkan objek tersebut.

10. Generalisasi dan Spesialisasi

Menunjukan hubungan antara kelas dan objek yang umum dengan kelas dan objek yang khusus. Misalnya kelas yang lebih umum (generalisasi) adalah kendaraan darat dan kelas khususnya (spesialisasi) adalah mobil, motor, dan kereta.

11. Komunikasi antar objek

Komunikasi antar objek dilakukan lewat pesan (message) yang dikirim dan satu objek ke objek lainnya.

12. Polimorfisme (*polymorphism*)

Kemampuan suatu objek untuk digunakan di banyak tujuan yang berbeda dengan nama yang sama sehingga menghemat baris program.

13. Package

Package adalah sebuah kontainer atau kemasan yang dapat digunakan untuk mengelompokkan kelas-kelas sehingga memungkinkan beberapa kelas yang bernama sama disimpan dalam package yang berbeda.

2.15 Pengujian Perangkat Lunak

Pengujian perangkat lunak adalah proses untuk mengevaluasi perangkat lunak baik secara manual maupun otomatis untuk menguji perangkat lunak memenuhi syarat atau belum, atau untuk menentukan perbedaan hasil yang diharapkan dengan hasil yang sebenarnya. Pelaksanaan pengujian perangkat lunak biasanya dibuat seiring membangun metodologi pembangunan lunak yang digunakan.

Pengujian perangkat lunak dikatakan berhasil jika pengujian yang dilakukan memiliki probabilitas yang tinggi untuk menemukan kesalahan (*error*) yang belum pernah ditemukan sebelumnya. [16]

Salah satu teknik pengujian blackbox adalah metode *equivalence partition class*. Metode *equivalence partition class* merupakan Merupakan test case yang ideal mengungkapkan kelas kesalahan, karena pada teknik ini berusaha mengungkapkan kelas-kelas kesalahan sehingga mengurangi jumlah total test case yang harus dikembangkan. Metode ini membagi domain input dari suatu program kedalam kelas - kelas data sehingga test case dapat diperoleh. Kelas data yang terbentuk disajikan sebagai kondisi input dalam kasus uji. Kelas merupakan himpunan nilai-nilai yang valid dan tidak valid. Desain test case partisi ekivalensi didasarkan pada evaluasi terhadap kelas ekivalensi untuk suatu kondisi input. [5]

2.16 Pengujian Akurasi

Confusion Matrix berisi informasi tentang klasifikasi aktual dan yang telah terprediksi yang dilakukan oleh sistem klasifikasi. Pada umumnya dievaluasi dengan menggunakan data matriks. Metode klasifikasi akan dievaluasi terutama pada bagian akurasi dari hasil klasifikasi. Akurasi sebuah klasifikasi berpengaruh terhadap performa dari suatu klasifikasi. Untuk melakukan analisa dapat digunakan confusion matrix, yaitu sebuah matrik dari prediksi yang akan dibandingkan dengan kelas yang asli dari data inputan. Berikut ini contoh tabel yang menunjukan confusion matrix untuk klasifikasi dua kelas.

Tabel 2.1 Confusion Matriks

		Kelas Prediksi	
		1	0
Kelas Sebenarnya		TP	FP
		FN	TN

Berdasarkan pada tabel 2.1, disimpulkan bahwa

True Posstive (TP), yaitu jumlah dokumen dari kelas 1 yang benar dan diklasifikasikan sebagai kelas 1.

True Negative (TN), yaitu jumlah dokumen dari kelas 0 yang benar diklasifikasikan sebagai kelas 0.

False Positive (FP), yaitu jumlah dokumen dari kelas 0 yang salah diklasifikasikan sebagai kelas 1.

False Negative (FN) yaitu jumlah dokumen dari kelas 1yang salah diklasifikasikan sebagai kelas 0.

Setiap kolom dari tabel confusion matrix merupakan contoh di kelas yang telah diprediksi, sedangkan setiap baris mewakili contoh di kelas yang sebenarnya. Setelah mendapatkan nilai untuk masing-masing kelas, selanjutnya adalah menghitung nilai akurasinya.

Persamaan perhitungan akurasi adalah sebagai berikut

$$Akurasi = \frac{TP + TN}{TP + FN + FP + FN} * 100 \quad (2.11)$$