BAB 2
TINJAUAN PUSTAKA

2.1 Teori Dasar Musik

Dalam seni musik. Dikenal istilah tangga nada. Istilah tersebut berisikan kumpulan nada-nada yang harmonis. Nada-nada yang harmonis ini terjadi karena ada aturan dibalik itu semua. Dibawah ini akan dijelaskan mengenai frekuensi dan nada, tangga nada, diantaranya tangga nada kromatik dan tangga nada diatonik, serta akor [4].

2.1.1 Frekuensi dan Nada

Frekuensi merupakan komponen dasar pembentuk suara. Frekuensi adalah kecepatan perubahan amplitude terhadap waktu. Suara muncul karena adanya getaran pada udara. Jumlah getaran pembentuk suara dimuatakan dalam frekuensi atau jumlah getaran dalam satu satuan waktu. Frekuensi dinyatakan dalam satuan Hertz (Hz). Misalkan dalam suatu suara memiliki frekuensi sebesar 440 Hz, maka berarti dalam satu detik terjadi 440 buah getaran.

Setiap nada memiliki frekuensi masing-masing. Secara internasional, nada A4 (A pada oktaf ke-4) memiliki frekuensi 440 Hz. Dan untuk mengetahui berapa frekuensi nada-nada lainnya dapat menggunakan persamaan:

$$fn = fo * an \tag{2.1}$$

fn adalah frekuensi nada dengan jarak-n yang dicarai, fo adalah frekuensi yang diketahui (A4= 440 Hz), an adalah $2^{s/12}$ (symbol S menunjukkan perbedaan semi-tone antara fn dan fo) [4].

2.1.2 Tangga Nada Kromatik

Tangga nada kromatik adalah kumpulan dari semua nada dalam musik, Dalam hal ini, tangga nada kromatik berarti nada dari setiap “warna” atau menyatakan frekuensi yang berbeda-beda sama halnya seperti cahaya. Dikarenakan

nada selalu berulang untuk tiap oktaf yang ada, maka istilah tangga nada kromatik" sering dipakai untuk ke-12 nada dari tiap oktaf [4]. Tabel berikut ini menunjukkan frekuensi dari ke-12 nada antara nada A pada 440 Hz dan nada A satu oktaf di atasnya.

Tabel 2.1 Tangga Nada Kromatik

Tangga Nada Kromatik			
Oktaf ke-3	**Frekuensi**	**Oktaf ke-4**	**Frekuensi**
A3	220,00 Hz	A4	440,00 Hz
A♯3	233,08 Hz	A♯4	466,16 Hz
B3	246,94 Hz	B4	493,88 Hz
C3	261,63 Hz	C4	523,25 Hz
C♯3	277,18 Hz	C♯4	554,37 Hz
D3	293,66 Hz	D4	587,33 Hz
D♯3	311,13 Hz	D♯4	622,25 Hz
E3	329,63 Hz	E4	659,26 Hz
F3	349,20 Hz	F4	698,46 Hz
F♯3	369,99 Hz	F♯4	739,98 Hz
G3	392,00 Hz	G4	784,00 Hz
G♯3	415,30 Hz	G♯4	830,60 Hz

Perbedaan antara dua buah *pitch* (nada) yang berdekatan disebut sebagai semiton. Meskipun ada 12 nada dalam 1 oktaf, tapi hanya 7 huruf pertama dari abjad yang dipakai untuk memberi nama pada nada, yaitu dari A sampai G. Kelima nada yang lain dalam tangga nada kromatik diberi nama dengan menempatkan tanda kres (♯) atau tanda mol (b) setelah notasi nada.

2.1.3 Tangga Nada Diatonis

Skala diatonik memiliki tujuh not yang berbeda dalam satu oktaf. Pada tangga nada diatonik di bagi dua yaitu [4]:

a. Mayor scales (Tangga nada Mayor)

   Skala mayor merupakan tangga nada yang mempunyai jarak antar nada dengan rumus interfal 1-1-½-1-1-1-½

b. Minor scales (Tangga nada Minor)

   Skala minor murupakan tangga nada yang mempunyai jarak antar nada

dengan interfal 1-½-1-1-½-1-1 (untuk tangga nada minor alami /melodis menurun).

Tangga nada minor sendiri dapat dibagi menjadi tiga:

1. Tangga Nada Minor Alami (melodis menurun):
   Interval 1-½-1-1-1-½
2. Tangga Nada Minor Melodis (melodis menaik):
   Interval 1-½-1-1-½-1-1
3. Tangga Nada Minor Harmonis:
   Interval 1-½-1-1-½-1-½

2.1.4 Akor

Akor atau *chord* adalah gabungan beberapa nada dasar yang dibunyikan secara serempak ataupun terpisah sehingga menghasilkan suara harmonis. Umumnya antara akor dan kunci adalah hal yang sama bagi kebanyakan orang awam, namun secara istilah kedua hal ini berbeda. Akor dalam istilah musik merujuk pada tiga nada dasar atau lebih yang dibunyikan baik secara serempak maupun terpisah. Sementara kunci atau key dalam not balok diatandai dengan notasi kres (#) yang berarti nada dinaikkan setengah dari nada dari nada dasarnya dan mol (b) yang berarti nada diturunkan setengah nada dari nada dasarnya [4].

2.1.4.1 Akor Mayor

Akor mayor biasanya dituliskan hanya berupa huruf capital seperti C, D, E, F, G, A, B. Mencari *chord* mayor dapat dilakukan dengan menggunakan nada ke 1-3-5 dari tangga nada C mayor, Contoh: akor C = Nada C – E – G [4].

2.1.4.2 Akor Minor

Akor minor biasanya dituliskan dengan menambahkan karakter ‘m’ setelah huruf capital seperti Cm, Dm,Em, Fm, Gm, Am, Bm. Ada pula yang menuliskan akor minor dengan huruf kecil saja seperti c, d, e, f, g, a, b. Apabila salah satu akor mayor sudah diketahui (misalnya C mayor), maka akor minornya yaitu (C minor)

dapat dengan mudah ditentukan yaitu dengan cara menurunkan nada yang ada ditengah sebanyak setengah interval. Contoh: Akor Cm = Nada C- D#/Eb – G [4].

2.2 Format File Audio *Wave*

WAV adalah format file audio standar Microsoft dan IBM untuk *personal computer* (PC), biasanya menggunakan pengkodean PCM (*Pulse Code Modulation).* WAV adalah data tidak terkompres sehingga seluruh sample audio disimpan semuanya di haridisk.

Secara umum data audio *digital* dari WAV memiliki karakteristik yang dapat dinyatakan dengan parameter-parameter berikut:

a. Laju sample (*sampling rate*) dalam sampel/detik, misalnya 22050 atau 44100 sampel/detik.
b. Jumlah bit tiap sample, misalnya 8 atau 16 bit.
c. Jumlah kanal (*channel*), yaitu 1 untuk mono dan 2 untuk *stereo*.

Biasanya laju sampel juga dinyatakan dengan satuan Hz atau kHz. Sebagai gambaran, data audio *digital* yang tersimpan dalam CD audio memiliki karakteristik laju sampel 44100 Hz, 16 bit persample, dan 2 kanal (stereo) yang berarti setiap satu detik suara tersusun dari 44100 sample, dan setiap sampel tersimpan dalam data sebesar 16-bit atau 2 byte. Laju sampel selalu dinyatakan untuk setiap satu kanal (*channel*). Jadi misalkan suatu data audio *digital* memiliki 2 kanal (*channel*) dengan laju sampel 8000 sampel/ detik, maka didalam setiap detiknya terdapat 16000 sampel [13].

2.3 *Preprocessing*

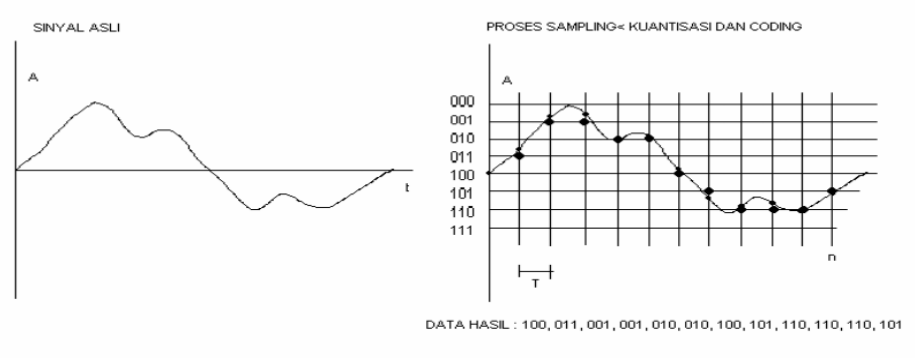
Preprocessing merupakan tahapan-tahapan yang dilakukan untuk mengkondisikan sinyal sebelum masuk kedalam tahap utama yaitu pengklasifikasian. Berikut adalah proses-prosesnya:

2.3.1 Konversi *Analog* Menjadi *Digital*

Signal – signal yang natural pada umumnya seperti *signal* suara merupakan *signal continue* dimana memiliki nilai yang tidak terbatas. Sedangkan pada

komputer, semua signal yang dapat diproses oleh komputer hanyalah *signal discrete* atau sering dikenal sebagai istilah *digital signal*. Agar *signal* natural dapat diproses oleh komputer, maka harus diubah terlebih dahulu dari data *signal continue* menjadi *discrete*. Hal itu dapat dilakukan melalui 3 proses, diantaranya sebagai berikut [17] :

1. Proses sampling adalah suatu proses untuk mengambil data *signal continue* untuk setiap periode tertentu. Dalam melakukan proses sampling data, berlaku aturan Nyquist, yaitu bahwa frekuensi sampling (sampling *rate*) minimal harus 2 kali lebih tinggi dari frekuensi maksimum yang akan di sampling. Jika signal sampling kurang dari 2 kali frekuensi maksimum signal yang akan di sampling, maka akan timbul efek *aliasing*. *Aliasing* adalah suatu efek dimana signal yang dihasilkan memiliki frekuensi yang berbeda dengan signal aslinya.
2. Proses kuantisasi adalah proses untuk membulatkan nilai data ke dalam bilangan-bilangan tertentu yang telah ditentukan terlebih dahulu. Semakin banyak *level* yang dipakai maka semakin akurat pula data *signal* yang disimpan tetapi akan menghasilkan ukuran data besar dan proses yang lama.
3. Proses pengkodean adalah proses pemberian kode untuk tiap-tiap data *signal* yang telah terkuantisasi berdasarkan *level* yang ditempati.



Gambar 2. 1 Proses Pembentukan Sinyal *Digital*

Pada Gambar 2.1 terjadi proses konversi sinyal *analog* menjadi sinyal *digital*, gambar sebelah kiri merupakan gambar sinyal asli yang masih berbentuk sinyal *analog* sedangkan gambar sebelah kanan merupakan gambar setelah melakukan

proses konversi sinyal analog menjadi sinyal *digital* yang telah melalui proses sampling, kuantisasi dan pengkodean. Pada gambar sebelah kanan didapatkan data hasil dalam bentuk kode yaitu 100, 011, 001, 001, 010, 010, 010, 100, 101, 110, 100 dan 101.

2.3.2 Normalisasi

Normalisasi adalah proses dimana membuat penskalaan pada nilai amplitudo tiap data sinyal sesuai skala yang diinginkan. Proses ini dilakukan agar nilai amplitudo pada tiap data sinyal yang akan diolah bernilai sama. Besarnya nilai amplitudo sinyal suara manusia saat melakukan pengucapan selau berbeda-beda, sehingga penskalaan nilai amplitudo sinyal terhadap acuan skala yang diinginkan sangat diperlukan [18].

$$X_{normalisas} = \frac{X_{masukan}}{\max(abs(X_{masukan}))} \quad (2.3)$$

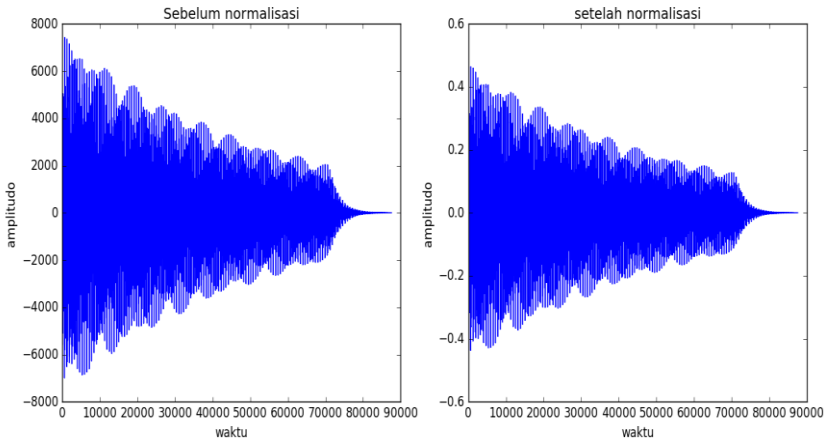
Keterangan:

$X_{normalisasi}$ = hasil data sinyal normalisasi

$X_{masukan}$ = amplitudo masukan

$\max(abs(X_{masukan}))$ = amplitudo maksimal masukan (16000)

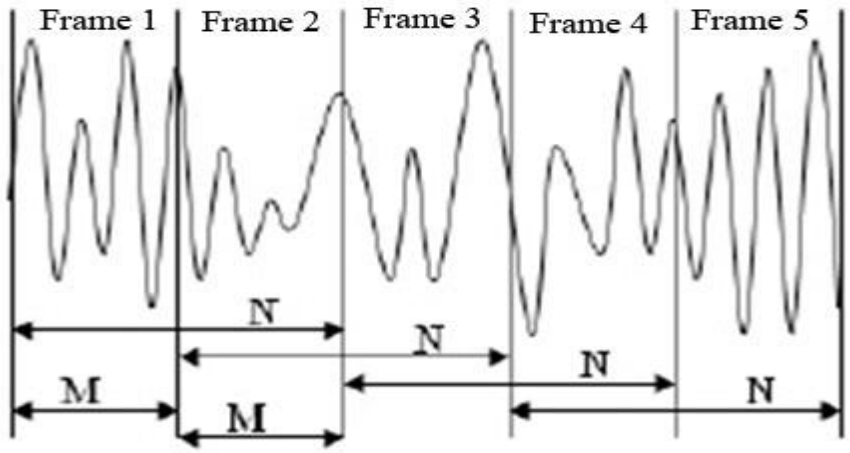
Proses normalisasi sinyal suara diperlihatkan pada gambar 2.2. Gambar pada sisi atas memperlihatkan sinyal yang akan dinormalisasi dan gambar pada sisi bawah memperlihatkan hasil normalisasinya. Dari gambar tersebut diperlihatkan berubahnya nilai amplitudo setelah skalanya diperbesar.



Gambar 2. 2 Contoh hasil proses normalisasi

2.3.3 *Frame Blocking*

Frame Blocking adalah proses dimana sinyal data masukan akan dibentuk dalm *frame-frame*. Dalam bentuk *frame* ini data akan lebih mudah untuk diketahui, sehinga tidak perlu memeriksa sinyal data scara keseluruhan secara langsung, Data dicek setiap *frame* sebesar nilai panjang *frame* yang telah ditentukan. Jika dalam pemeriksaan sinyal data melebihi maka akan mengalami overlap, dimana akan sinyal data yang berlebih tersebut akan dipotong sebesar nilai overlap [19].



Gambar 2. 3 *Frame Blocking*

Keterangan:

Frame1, frame2, frame3, frame4,frame5 adalah sinyal suara yang di *frame blocking*, dimana M panjang pipa *frame* dan N overlapping tiap *frame*-nya.

Jumlah *frame* = ((Total sinyal amplitudo/M)+N+1) (2.4)

Keterangan :

Jumlah *frame*	= Total *frame* berdasarkan pembagian panjang *frame* dengan *Overlapping*
M	= Panjang *frame*
N	= *Overlapping* (M/2)

2.3.4 *Windowing*

Proses *framing* dapat menyebabkan terjadinya kebocoran spektral (*spectral leakage*) atau *aliasing. Aliasing* adalah *signal* baru dimana memiliki frekuensi yang berbeda dengan *signal* aslinya. Efek ini dapat terjadi karena rendahnya jumlah *sampling rate*, ataupun karena proses *frame blocking* dimana menyebabkan *signal* menjadi *discontinue*. Untuk mengurangi kemungkinan terjadinya kebocoran spektral, maka hasil dari proses *framing* harus melewati proses *window* [17].

Sebuah fungsi *window* yang baik harus menyempit pada bagian *main lobe* dan melebar pada bagian *side lobe*-nya. Berikut ini adalah representasi dari fungsi *window* terhadap *signal* suara yang diinputkan.

$$x(n) = x_i(n)w(n) \quad (2.5)$$

Keterangan :

n = 0,1,…,N-1

$x(n)$ = nilai sampel *signal* hasil *windowing*

$x_i(n)$ = nilai sampel dari *frame signal* ke i

$w(n)$ = fungsi *window*

N = *frame size*

Ada banyak fungsi *window*, namun yang paling sering digunakan dalam aplikasi *speaker recognition* adalah *hamming window*. Fungsi *window* ini menghasilkan *sidelobe level* yang tidak terlalu tinggi (kurang lebih -43 dB), selain itu *noise* yang dihasilkan pun tidak terlalu besar.
Fungsi *Hamming window* adalah sebagai berikut :

$$w(n) = 0.54 - 0.46\, cos\, \frac{2\pi n}{M-1} \tag{2.6}$$

Keterangan :
n = 0,1,...,M-1
M = panjang *frame*

2.3.5 Operasi *Powering*

Operasi powering dilakukan untuk menemukan *peak* dari tiap *frame blocking*. Pada proses ini sinyal input diproses dengan mengambil *peak-peak* dominan. Sebelumnya, sinyal akan mengalami *powering* terlebih dahulu dengan melakukan pengkuadratan pada sinyal, dengan persamaan [16] :

$$P = \sum_{i=0}^{i=N} \left| {X_i}^2 \right| \tag{2.7}$$

Keterangan :
P = *power* sinyal (dB)
|Xi| = norm *Frame* ke-i (dB)
N = Jumlah sampel per *frame*
i = sampel ke-i

Powering bertujuan untuk mengetahui letak awal dan akhir dari suatu sinyal yang berada di suatu *frame*. Dari nilai *power* didapatkan suatu nilai rata-rata dengan menambahkannya dengan standar deviasi, maka didapatkan nilai awal dan akhir dari suatu *frame*. Standar deviasi merupakan penyimpangan data standar dari data

yang ada. Jika nilai data mendekati nilai rata-rata, maka standar deviasinya lebih kecil (mendekati nol) dan sebaliknya. Ditunjukkan dengan persamaan [1].

$$Standar\ Deviasi = \sqrt{\frac{n\ \sum {P_i}^2 - (\sum P_i)^2}{n(n-1)}} \quad (2.8)$$

Keterangan :

P_i = *Frame* ke-i

n = jumlah frame

Sinyal dengan nilai power berada diatas standar deviasi, diambil sebagai awal dan akhir sinyal serta dianggap sebagai suatu suara.

2.3.6 *Thresholding*

Pada proses ini dilakukan pengambilan *peak* dominan dari sinyal dengan terlebih dahulu melakukan *powering* dan penentuan standar deviasi yang berguna sebagai pendeteksi awal dan akhir sinyal dalam *frame*. Setelah itu, sinyal dibagi menjadi *frame-frame* untuk mempermudah pemrosesan sinyal. Pengambilan sample setiap *frame* diambil dalam waktu milisecond (ms) [16].

2.3.7 *Fast Fourier Transform*

DFT (*Discrete Fourier Transform*) merupakan perluasan dari transformsi *fourier* yang berlaku untuk *signal-signal* diskrit dengan panjang yang terhingga. Semua *signal* periodik terbentuk dari gabungan *signal-signal* sinusoidal yang menjadi satu yang dapat dirumuskan sebagai berikut.

$$F[k] = \sum_{n=0}^{N-1} f[n] e^{-\frac{j2\pi nk}{N}}, 0 \leq k \leq N-1 \quad (2.7)$$

N = jumlah sampel yang akan diproses ($N \in$ N)

$f[n]$ = nilai sampel *signal*

K = variable frekuensi discrete, dimana akan bernilai ($k = {}^{N}/_{2}\ , k\ \in$ N)

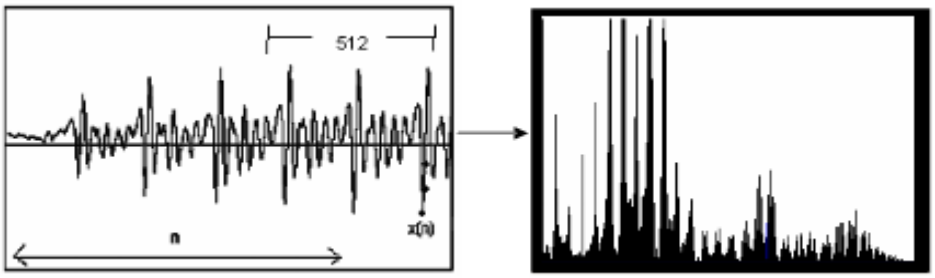
Bentuk yang digunakan dalam FFT adalah sebagai berikut

$$F(k) = \sum_{n=1}^{N} \left(f_{(n)} \cos \frac{2\pi nkT}{N} \right) - j \sum_{n=1}^{N} \left(f_{(n)} \sin \frac{2\pi nkT}{N} \right) \quad (2.8)$$

Untuk menghitung hasil FFT digunakan rumus

$$|f(u)| = [R^2 + I^2]^{\frac{1}{2}} \quad (2.9)$$

Dengan persamaan 2.8 suatu *signal* suara dalam domain waktu dapat kita cari frekuensi pembentuknya. Hal inilah tujuan penggunaan analisa *fourier* pada data suara, yaitu untuk merubah data dari domain waktu menjadi data spektrum di domain frekuensi. Untuk pemrosesan *signal* suara, hal ini sangatlah menguntungkan karena data pada domain frekuensi dapat diproses dengan lebih mudah dibandingkan data pada domain waktu, karena pada domain frekuensi, keras lemahnya suara tidak seberapa berpengaruh [17].



Gambar 2. 4 Domain Waktu menjadi Domain Frekuensi

Pada Gambar 2.4 dapat dilihat bahwa gambar sebelah kiri menunjukan gambar domain waktu dan gambar sebelah kanan menunjukan gambar domain frekuensi.

Untuk mendapatkan spektrum dari sebuah *signal* dengan DFT diperlukan N buah sampel data berurutan pada domain waktu, yaitu x[m] sampai x[m+N-1]. Data tersebut dimasukkan dalam fungsi DFT maka akan menghasilkan N buah data. Namun karena hasil dari DFT adalah simetris, maka hanya N/2 data yang diambil sebagai spektrum.

2.4 Fuzzy c-means Clustering

Metode *Fuzzy c-means* pertama kali ditemukan oleh Jim Bezdek pada tahun 1981, meode Fuzzy C- Means merupakan salah satu teknik clustering yang mana keberadaan tiap titik data ditentukan oleh derajat keanggotaan (partisi). Teori Fuzzy keanggotaan sebuah data tidak deiberikan nilai secara tegas dengan nilai 1(menjadi anggota dan 0 (tidakmenjadi anggota), melainkan dengan nilai derjat keanggotaan yang jangkauan nilainya 0 sampai 1. Nilai keanggotaan suatu data dalam sebuah himpunan menjadi 0 sama sekali tidak menjadi anggota dan menjadi 1 jika menjadi anggota secara penuh dalam suatu himpunan. Umumnya nilai keanggotaannya antara 0 dan 1. Semakin tinggi nilai keanggotaannya maka semakin tinggi derajat keanggotaannya dan semakin kecil maka semakin rendah derajat keanggotaannya [8].

Konsep *fuzzy c-means* ini pertama kali adalah menentukan pusat cluster, yang menandai lokasi rata-rata tiap cluster. Kondisi awal pusat cluster ini belum akurat. Tiap-tiap titik data memiliki derajat keanggotaan untuk tiap-tiap cluster dengan cara memperbaiki pusat cluster dan derajat keanggotaan tiap-tiap titik data secara berulang, maka akan dapat dilihat bahwa pusat cluster akn bergerak menuju lokasi yang tepat. Perulangan ini didasarkan pada minimasi fungsi obyektif yang menggambarkan jarak dari titik data yang diberikan kepusat cluster yang erbobot oleh derajat keanggotaan titik data tersebut [7].

Algoritma Fuzzzy C-Means adalah sebagai berikut:

1. Penentuan pusat cluster yang menandai lokasi rata-rata untuk tiap cluster dengan kondisi awal tidak akurat.
2. Tiap data memiliki derajat keanggotaan untuk masing-masing cluster
3. Perulangan yang didasarkan pada minimasi fungsi obyektif, pusat cluster dan nilai keanggotaan diperbaiki sehingga lokasi cluster bisa berada pada posisi yang benar.

Algoritma *Fuzzy c-means* didasarkan pada fungsi objektif yang diformulasikan dalam persaman sebagai berikut [10]

1. Inputkan data yang akan dikelompokkan, yaitu X berupa matriks dengan

ukuran n x m(n= jumlah sampel data, m= atribut setiap data). Xij data sampel ke-i (i = 1,2,3,..,n), atribut ke-j (j = 1,2,3,..,m).

2. Tentukan julmlah cluster (c). Pangkat untuk matriks partisi (w), maksimum iterasi (MaxIter), error terkecil yang diharapkan, fungsi objektif awal (Po = 0), dan iterasi awal (t =1).
3. Bangkitkan bilangan random $\mu_{ik}, i = 1,2,..,n; k = 1,2,\ldots,c$ sebagai elemen matriks partisi awal *U*
4. Hitung pusat cluster ke-*k:* $V_{kj,}$ dengan *k* = 1,2,…,c, Dan *j* =1,2,..,m, menggunakan persamaan berikut:

$$V_{kj} = \frac{\sum_{i=1}^{n}((\mu_{ik})^{w} * X_{ij})}{\sum_{i=1}^{n}(\mu_{ik})^{w}} \quad (2.10)$$

Keterangan:

V_{kj} = pusat *cluster* ke-*k* untuk atribut ke-*j*

μ_{ik} = derajat keanggotaan untuk data sampel ke-*i* pada cluster ke-*k*

x_{ij} = data ke-*i*, atribut ke-*j*

5. Hitung fungsi objektif pada iterasi ke-t menggunakan persaman berikut

$$Pt_{ik} = \sum_{i=0}^{n}\sum_{k=0}^{c}\left(\left[\sum_{j=0}^{m}(X_{ij} - V_{kj})^{2}\right](U_{ik})^{w}\right) \quad (2.11)$$

Keterangan:

V_{kj} = pusat *cluster* ke-*k* untuk atribut ke-*j*

U_{ik} = derajat keanggotaan untuk data sampel ke-*i* pada cluster ke-*k*

x_{ij} = data ke-*i*, atribut ke-*j*

Pt_{ik} = fungsi objektif pada iterasi ke-T, data ke-i, dan kelas ke-k

6. Hitung perubahan matriks partisi menggunakan persamaan berikut:

$$\mu_{ik} = \frac{[\sum_{j=1}^{m}(X_{ij} - V_{kj})^{2}]^{\frac{1}{w-1}}}{\sum_{k=1}^{c}[\sum_{j=1}^{m}(X_{ij} - V_{kj})^{2}]^{\frac{1}{w-1}}} \quad (2.12)$$

Dengan I = 1,2,3,..,n, dan k = 1,2,..,c

V_{kj} = pusat *cluster* ke-*k* untuk atribut ke-*j*

μ_{ik} = derajat keanggotaan untuk data sampel ke-*i* pada cluster ke-*k*

x_{ij} = data ke-*i*, atribut ke-*j*

7. Cek kondisi berhenti:

   Jika : ($|Pt - Pt - I| < \xi$) atau (t > MaxIter) maka akan berhenti, jika tidak :

   $T = t + I$, ulangi langkah ke-4

2.5 Euclidean Distance

Euclidean distance adalah sebuah metode yang digunakan untuk mengukur jarak (*distance*). *Euclidean distance* sebenarnya merupakan generalisasi dari teorema phytagoras. *Euclidean distance* diformulasikan pada persamaan berikut:

$$d(x, y) = \sqrt{\sum_{i=1}^{r}(x_i - y_i)^2} \quad (2.10)$$

d = jarak antara x dan y

r = jumlah fitur dalam vektor

x_i, y_i = nilai fitur ke i dari x dan y

Jarak minimal yang mungkin antara dua vektor data adalah 0. Selain itu, jarak dari *x* ke *y* akan sama dengan *y* ke *x*, $d(x, y) = d(y, x)$

2.6 Pemrograman Berorientasi Objek

Pemograman berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya. Menggunakan pendekatan berorientasi objek akan memandang sistem yang akan dikembangkan sebagai suatu kumpuln objek yang berkorespondensi dengan objek dunia nyata.

Dalam rekayasa perangkat lunak, konsep pendekatan berorientasi objek dapat diterapkan pada tahap analisis, perancangan, pemograman, dan pengujian perangkat lunak. Berikut adaah beberapa konsep dasar yang harus dipahami pemograman berorientasi objek [6] :

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 dari kelas tersebut. Sebuah kelas akan mempunyai sifat (atribut),

kelakuan (operasi/metode), hubungan (*relationship*) dan arti. Suatu kelas dapat diturunkan dan kelas semula dapat diwariskan ke kelas yang baru.

2. Objek (*Object)*

Objek adalah abstraksi dan sesuatu yang mewakili dunia nyata seperti benda, manusia, satuan organisasi, tempat, kejadian, strutur, 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*)

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

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
30 dalam kelas objek. Atribut secara individual oleh sebuah objek, misalnya berat, jenis, nama, dan sebagainya.

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. Enkapulasi (*Encapsulation*)

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

7. Pewarisan (*Inheritance*)

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

8. Antarmuka (*Interface*)

Antarmuka atau interface sangat mirip dengan kelas, tetapi tanpa atribut kelas dan tanpa memiliki metode yang dideklarasikan. Antarmuka biasanya digunakan agar kelas lain tidak langsung mengakses ke suatu kelas.

9. *Reusability*

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

10. Generalisasi dan Spesialisasi

Menunjukkan 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 dibanyak tujuan yang berbeda dengan nama yang sama sehingga menghemat baris program.

13. *Package*

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

2.7 Python

Python adalah sebuah Bahasa pemrograman dinamis yang sering digunakan dalam pengembangan aplikasi pada berbagai domain. Menurut Hall (2005, p10) Python pada dasarnya memiliki style yang serupa dengan *pseudocode*, Perbedaannya hanyalah python dapat dijalankan di komputer dan menampilkan hasil. Bahasa Python mudah dimengerti (serupa dengan Bahasa Inggris), dan tidak perlu compiling, maka dari itu kita dapat menjalankan *script* Python begitu kita menyimpannya tanpa memerlukan langkah lanjutan.

2.8 Unified Modelling Language (UML)

UML singkatan dari *Unifed Modeling Language* yang berarti bahasa pemodelan standar. UML merupakan bahasa standar untuk merancang dan mendokumentasikan perangkat lunak dengan cara berorientasi objek. Ada beberapa diagram yang digunakan proses pembuatan perangkat lunak berorientasi objek diantaranya, *use case diagram, activity diagram, class diagram* dan *sequence diagram* [5].

2.8.1 *Use Case Diagram*

Use case diagram merupakan pemodelan untuk tingkah laku (*behavior*) pada sistem yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat. *Use case diagram* digunakan untuk mengetahui fungsi apa saja yang terdapat pada sistem. Terdapat dua hal utama yang diperlukan dalam pembentukan suatu *use case diagram* yaitu aktor dan *use case*.

1. Aktor merupakan orang, benda maupun sistem lain yang berinteraksi dengan sistem yang akan dibangun.
2. *Use Case* merupakan fungsionalitas atau layanan yang disediakan oleh sistem sebagai unit-unit yang saling bertukar pesan antar unit atau actor [5].

2.8.2 *Activity diagram*

Activity diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem, proses bisnis atau menu yang ada pada perangkat lunak. Setiap *use case* yang telah dibentuk digambarkan aktivitasnya dalam *activity diagram*, mulai dari peran aktor, peran sistem, dan *decision*. *Activity diagram* juga banyak digunakan untuk mendefinisikan hal-hal berikut [5]:

1. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem.
2. Urutan atau pengelompokan tampilan dari sistem / *user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan tampilan antarmuka.

3. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.
4. Rancangan menu yang ditampilkan pada perangkat lunak.

2.8.3 *Class diagram*

Class diagram menggambarkan interaksi dan relasi antar kelas yang ada di dalam suatu sistem. Kelas memiliki atribut dan metode. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas. Metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas. Atribut dan metode dapat memiliki salah satu sifat sebagai berikut [5]:

1. *Private,* tidak dapat dipanggil dari luar kelas yang bersangkutan.
2. *Protected,* hanya dapat dipanggil oleh kelas yang bersangkutan dan anak-anak yang mewarisinya.
3. *Public,* dapat dipanggil oleh siapa saja.

Class diagram menggambarkan relasi atau hubungan antar kelas dari sebuah sistem. Berikut ini beberapa gambaran relasi yang ada dalam *class diagram*:

1. *Association*

   Hubungan antar *class* yang statis. *Class* yang mempunyai relasi asosiasi menggunakan *class* lain sebagai atribut pada dirinya.
2. *Aggregation*

   Relasi yang membuat *class* yang saling terikat satu sama lain namun tidak terlalu berkegantungan.
3. *Composition*

   Relasi agregasi dengan mengikat satu sama lain dengan ikatan yang sangat kuat dan saling berkegantungan.
4. *Dependency*

   Hubungan antar *class* dimana *class* yang memiliki relasi *dependency* menggunakan *class* lain sebagai atribut pada *method*.
5. *Realization*

   Hubungan antar *class* dimana sebuah *class* memiliki keharusan untuk mengikuti aturan yang ditetapkan *class* lainnya.

2.8.4 *Sequence diagram*

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan *sequence diagram* maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

Banyaknya *sequence diagram* yang harus digambar adalah sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada *sequence diagram* sehingga semakin banyak *use case* yang didefinisikan maka *sequence diagram* yang harus dibuat juga semakin banyak. Penomoran pesan berdasarkan urutan iteraksi pesan. Penggambaran letak pesan harus berurutan, pesan yang lebih atas dari lainnya adalah pesan yang berjalan terlebih dahulu [5].

2.9 Pengujian

Pengujian yang dilakukan pada penelitian ini menggunakan pengujian *confussion matrix* dan pengujian validitas *fuzzy c-means* sebagai berikut:

2.9.1 Pengujian Black Box

Pengujian *Black Box* berfokus kepada persyaratan fungsional perangkat lunak (*software*) yang dibuat. *Black box Testing* (pengujian kotak hitam) yaitu menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksud untuk mengetahui apakah fungsi-fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan [14].

Pengujian Black Box dilakukan dengan membuat kasus uji yang bersifat mencoba semua fungsi dengan memakai perangkat lunak apakah sesuai dengan spesifikasi yang dibutuhkan. Pengujian Black Box berusaha menemukan kesalahan dalam kategori sebagai berikut.

1. Fungsi- fungsi yang tidak benar atau hilang

2. Kesalahan *interface*
3. Kesalahan dalam struktrur data atau akses *database* eksternal
4. Kesalahan kinerja
5. Inisialisasi dan kesalahan terminasi

2.9.2 Confussion Matrix

Untuk mengevaluasi performa dari hasil proses klasifiksi dapat ditampilkan pada *confussion matrix. Confussion Matrix* berisi tentang klasifikasi actual dan yang telah diprediksi yang dilakukan oleh sebuah system klasifikasi. Kinerja sebuah system klasifikasi umumnya dievaluasi dengan menggunakan data dalam matriks. Tabel 2.2 berikut menunjukan contoh confussion matrix untuk klasifikasi masalah biner dua kelas [15].

Tabel 2.2 *Confussion Matrix*

		Predicted Class	
		Kelas = 1	**Kelas = 0**
Actual Class	**Kelas = 1**	*f11*	*f10*
	Kelas = 0	*f01*	*f00*

Berdasarkan isi confussion matrix, maka dapat diketahui jumlah data dari masing-masing kelas yang diprediksi secara benar yaitu (*f11 + f00*) dan data yang diklasifikasikan salah yaitu (*f10 + f01*). Untuk menghitung akurasi digunakan formula sebagai berikut :

$$Akurasi = \frac{f11+f00}{f11+f10+f01+f00} \quad (2.11)$$

2.9.3 Pengujian Validitas Fuzzy C-Means

Metode pengelompokkan yang menggunakan konsep konsep *fuzzy* merupakan konsep yang sebuah data bisa menjadi anggota di semua cluster dengan nilai derajat eanggotaan yang dimilikinya. Semakin tinggi nilai derajat keanggotaan pada sebuah cluster maka semakin besar kecendrungannya menjadi anggot acluster

tersebut. Beberapa metode validitas fuzzy c-means yang digunakan adalah sebagai berikut.

2.9.3.1 Partition Coefficient Index (PCI)

Bezdek (1981) mengusulkan validitas dengan menghitung koefisien partisi atau *partition coefficient* (PC) sebagai evaluasi nilai keanggotaan data pada setiap cluster. Nilai PC Index (PCI) hanya mengevaluasi nilai keanggotaan, tanpa memandang nilai vektor (data) yang biasanya mengandung informasi geometrik (sebaran data). Nilainya dalam rentang [0,1], nilai yang semakin besar (mendekati 1) mempunyai arti bahwa kualitas cluster yang didapat semakin baik. Berikut formula untuk menghitung PC Index [7].

$$PCI = \frac{1}{N}\left(\sum_{i=1}^{N}\sum_{j=1}^{K} {u_{ij}}^2\right) \quad (2.12)$$

N adalah jumlah data dalam set data, K merupakan jumlah cluster, sedangkan u_{ij} menyatakan nilai keanggotaan data ke-i pada cluster ke-j.

2.9.3.2 Partition Entropy Index (PEI)

Bezdek (1974a,b) mengusulkan validitas dengan menghitung entropi partisi atau *partition entropy* (PE). Nilai PE Index (PEI) mengevaluasi keteracakan data dalam cluster. Nilainya dalam rentang [0,1], nilai yang semakin kecil (mendekati 0) mempunyai arti bahwa kualitas cluster didapat semakin baik. Berikut formula untuk menghitung PEI [7].

$$PEI = -\frac{1}{N}\left(\sum_{i=1}^{N}\sum_{j=1}^{K} u_{ij}\ x\ log_2 u_{ij}\right) \quad (2.13)$$

N adalah jumlah data dalam set data, K merupakan jumlah cluster, sedangkan u_{ij} menyatakan nilai keanggotaan data ke-i pada cluster ke-j.

2.9.3.3 Modification Partition Coefficient Index (MPCI)

Metode PCI dan PEI memiliki kecendrungan monotonik terhadap K. Modifikasi nilai PCI (MPCI) dilakukan oleh dave (1996) terhadap kecendrungan monotonik tersebut. Formula yang digunakan sebagai berikut.

$$MPCI = 1 - \frac{K}{K-1}\left(1 - PCI\right) \quad (2.14)$$

Nilai MPCI yang didapat adalah $0 \leq MPCI \leq 1$. Nilai MPCI ekuivalen dengan Non-Fuzzines Index (NFI).

2.10 Tools Pembangunan Perangkat Lunak

Adapun beberapa perangkat lunak pendukung dalam pembuatan aplikasi pengenalan akor ini diantaranya :

2.10.1 *Spyder*

Spyder adalah *Scientific Python Developmen Environment* yang merupakan salah satu IDE (*Integrated Development Environment*) dari *Python.* Spyder mempunyai ciri komputasi numerik yang didukung oleh IPython (*enhanced interactive Python interpreter*) dan pustaka popular pada *python* seperti NumPy (linear algebra), sciPy (*Signal* dan *Image processing*) atau matpLotLib (*interactive 2D/3D plotting*) [11].

2.10.2 *Qt Designer*

Qt Designer adalah Qt tool untuk mendesain dan membangun *Graphical User Interface* (GUI). Qt Designer dapat mendesain widget, dialog atau tampilan utama yang lengkap hanya dengan drag dan drop sederhana. Qt Designer memiliki kemampuan preview desain untuk memastikan desain sesuai dengan yang diinginkan. Qt Designer menggunakan XML.ui *file* untuk menyimpan desain. Qt memasukkan uic utility untuk menghasilkan kode C++ yang membuat user interface nya. Qt juga memasukkan QUILoader class yang berfungsi sebagai aplikasi untuk load ui *file* dan membuat hubungan user interface secara dinamis [12].