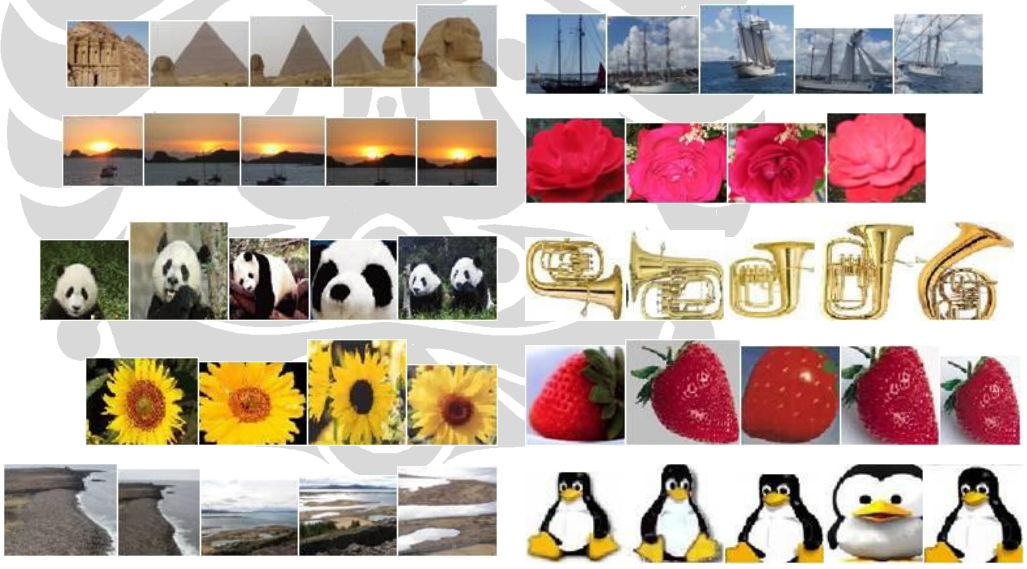
BAB III
DATASET DAN RANCANGAN PENELITIAN

Pada bab ini dijelaskan tentang dataset citra yang digunakan dalam penelitian ini serta rancangan untuk melakukan penelitian.

3.1 DATASET PENELITIAN

Penelitian ini menggunakan dataset yang diunduh dari INRIA (Inria, 2009) dan Coblitz (Coblitz, 2009) yang merupakan basisdata citra dengan format JPG dalam jumlah besar dengan komponen warna yang beragam. Sebelum melalui tahapan-tahapan dalam proses *image clustering*, dilakukan *cropping* dan *resizing* citra agar mudah dikomputasi yaitu dengan ukuran rata-rata per citra adalah sekitar 80x80. Contoh citra yang digunakan dalam eksperimen dapat dilihat pada Gambar 3.1.



Gambar 3.1 Contoh Citra yang digunakan

Citra yang dikoleksi adalah citra yang telah dipilih dengan 20 kategori berbeda, yaitu panda, padang pasir, laut, *sunset*, bunga matahari, mawar merah, bunga melati, buah anggur, kuda, dan sebagainya. Kategori ini dipilih berdasarkan komponen warna yang terkandung didalamnya karena tujuan penelitian ini adalah melakukan *clustering* berdasarkan komponen warna. Masing-masing kategori terdiri atas lebih kurang 80 citra, jadi total keseluruhan citra adalah 1600 citra. Tabel 3.1 menunjukkan kategori citra serta jumlah maksimum dari tiap kategori.

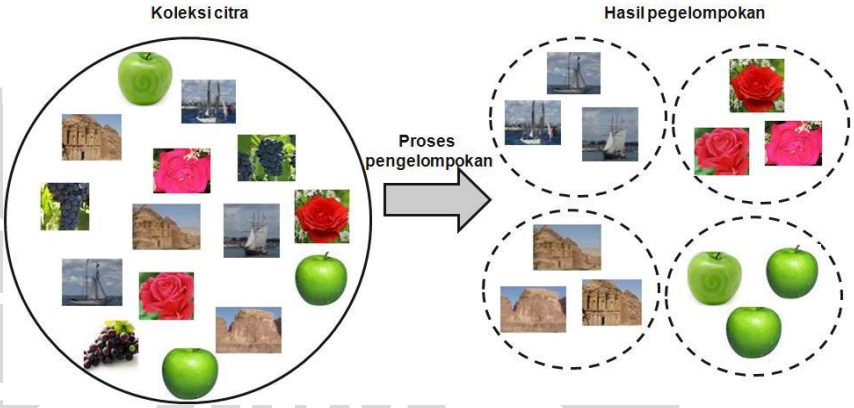
Tabel 3.1: 20 Kategori Citra

No	*Cluster*	Jumlah citra
1	Panda	80
2	Padang pasir	80
3	Laut	80
4	*Sunset*	80
5	Bunga matahari	80
6	Mawar merah	80
7	Bunga melati	80
8	Buah anggur	80
9	Kuda	80
10	Ikan hias	80
11	Pisang	80
12	Bangunan	80
13	Padang rumput	80
14	Terompet	80
15	Apel	80
16	Harimau	80
17	Pantai	80

18	Kembang api	80
19	Pinguin	80
20	Stroberi	80

3.2 RANCANGAN PENELITIAN

Penelitian yang dilakukan pada pengelompokan citra berdasarkan komposisi warna ini secara garis besar dapat dilihat pada Gambar 3.2.

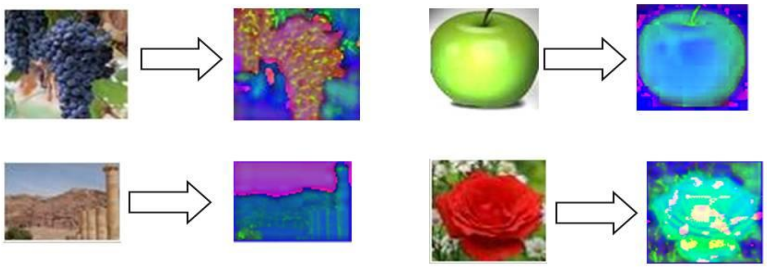


Gambar 3.2. *Image Clustering* Berdasarkan Komposisi Warna

Pada Gambar 3.2 terlihat pengelompokan citra dengan hasil yang optimal, yaitu citra dengan kategori atau jenis yang sama berada pada satu kelompok yang sama. Akan tetapi, tidak jarang terjadi pengelompokan yang tidak optimal karena beberapa sebab dan dalam penelitian ini ingin diperlihatkan seberapa baik proses pengelompokan citra yang menggunakan komposisi warna.

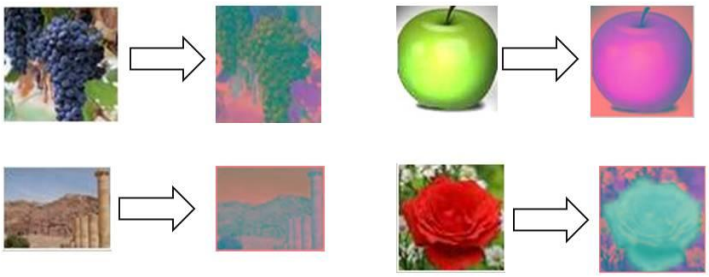
Proses pengelompokan citra ini terbagi atas 3 tahap, yaitu konversi ruang warna, representasi citra atau *feature extraction*, kemudian metode *clustering.* Pada Gambar 1.1 ditunjukan gambaran penelitian secara keseluruhan. Berikut ini adalah langkah-langkah pengelompokan citra.

1. Sekumpulan citra dikonversi ke ruang warna yang diinginkan



Gambar 3.3 Contoh Hasil Konversi Citra dari RGB ke HSV

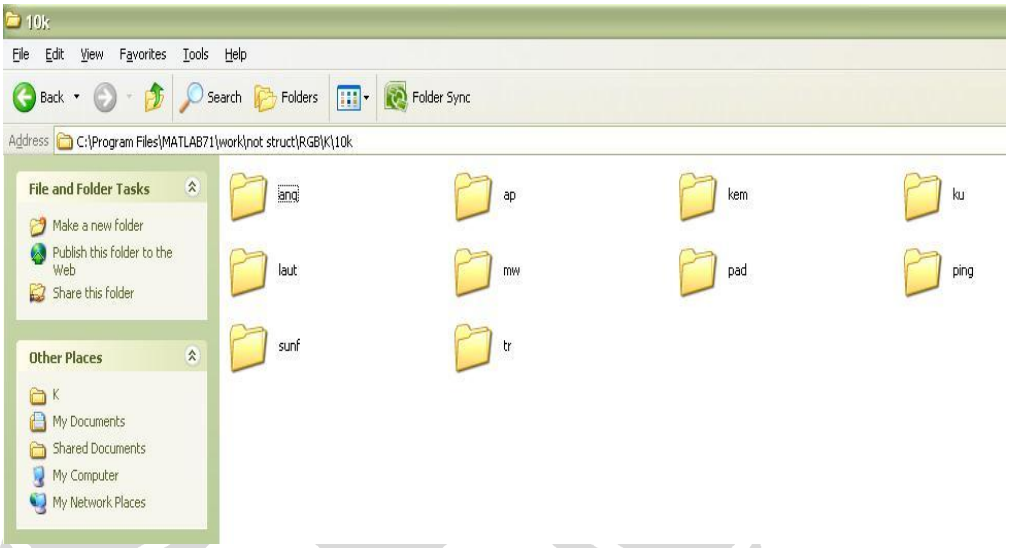
Gambar 3.3 merupakan contoh beberapa citra RGB yang kemudian di konversi ke ruang warna HSV. Citra disebelah kiri tanda panah adalah citra RGB, sedangkan citra disebelah kanan tanda panah adalah citra hasil konversi ke HSV. Gambar 3.4 berikut adalah contoh konversi citra dari ruang warna RGB ke L*a*b*.



Gambar 3.4 Contoh Hasil Konversi Citra dari RGB ke L*a*b*

2. Lakukan *feature extraction* pada tiap citra hasil konversi dengan menggunakan GMM atau histogram yang menghasilkan vektor yang merupakan ciri dari setiap citra asli.
3. Terapkan metode *clustering* (*AHC* atau *K-Means*) pada sekumpulan vektor hasil *feature extraction.*

4. Hasil *clustering* di salin ke dalam direktori yang berjumlah k direktori sesuai dengan jumlah kelompok citra yang diinginkan.
5. Pemberian label direktori yang merupakan label kelompok citra. Kategori citra yang paling banyak dalam direktori adalah label dari kelompok.



Gambar 3.5 Contoh Pelabelan Direktori Citra

Gambar 3.5 menunjukan contoh pemberian label pada direktori dimana label tersebut diberikan berdasarkan jumlah kategori citra dominan yang ada pada satu direktori. Misalnya direktori ‘ang’ didominasi oleh citra anggur, pada direktori ‘tr’ didominasi oleh citra trompet, dan seterusnya.

6. Hitung jumlah citra dalam suatu kategori sesuai dengan label direktori. Inilah yang merupakan citra yang benar yang masuk ke dalam pengelompokan yang tepat.

Gambar 3.6 Contoh Direktori Berlabel 'Pinguin'

Gambar 3.6 merupakan direktori citra pinguin yang kemudian dihitung jumlah citra pinguin yang ada pada direktori tersebut.

Seperti yang telah dijelaskan sebelumnya, awalnya citra dibaca kemudian dilakukan konversi dari ruang warna RGB ke ruang warna HSV dan L*a*b*. Mengenai proses konversi ruang warna dijelaskan pada bagian **3.3 KONVERSI RUANG WARNA.**

Setelah tahapan konversi ruang warna, dilakukan *feature extraction* yang didalam penelitian ini menggunakan metode *Gaussian Mixture Model (GMM)* dan histogram. Pada *GMM*, diterapkan distribusi *Gaussian* pada setiap kanal warna, misalnya merah, hijau, biru pada ruang warna RGB. Pada kanal warna merah dilakukan GMM dengan inisial kelompok piksel adalah 10, kemudian piksel berdekatan nilainya di kumpulkan menjadi satu ke dalam sebuah kelompok. Tiap piksel yang berdekatan nilainya berada pada kelompok yang berbeda dengan piksel yang memiliki nilai berbeda yang disebut sebagai *homogenous region*. Sebenarnya, setiap piksel tidak benar-benar dipindahkan untuk berkumpul pada suatu kelompok, tetapi hanya berupa probabilitas tiap piksel

untuk masuk ke dalam salah satu kelompok yang menggunakan *GMM*. Jadi, sifat *GMM* ini adalah *fuzzy*, dimana satu piksel tidak berada tepat pada satu *cluster*, tetapi berada pada beberapa *cluster* dengan probabilitas yang berbeda-beda pada tiap *cluster*-nya.

Sedangkan dengan metode histogram, *feature space* diperoleh dengan menghitung distribusi warna dari setiap piksel pada citra dalam interval tertentu. Dalam hal ini, setiap warna dipengaruhi oleh ruang warna yang digunakan. Proses representasi citra dijelaskan pada **3.4 *GAUSSIAN MIXTURE MODEL* DAN HISTOGRAM.**

Sekumpulan *homogenous region* hasil *GMM* dan hasil dari histogram yang disebut sebagai *feature space* dari sebuah citra akan digunakan sebagai parameter untuk melakukan proses *clustering* yang menggunakan metode *Agglomerative Hierarchical Clustering (AHC)* dan *K-Means*. Proses *clustering* yang menggunakan *AHC* diawali dengan pengukuran jarak antara satu *feature space* dengan *feature space* yang lain dengan menggunakan metode *Euclidean distance*. Setelah itu, dilakukan *clustering* dengan menggunakan *average link*. Skema ini dianggap skema yang paling baik dalam *AHC* walaupun memerlukan waktu komputasi yang lebih tinggi dibanding dengan skema *single, complete*, ataupun *centroid*. Proses *clustering* dengan K-Means juga menggunakan pengukuran jarak *Euclidean.* Untuk lebih jelasnya dapat dilihat pada bagian **3.5 PROSES *CLUSTERING*.**

Setelah proses *clustering* berakhir, dilakukan evaluasi hasil *clustering* yang dijelaskan pada bagian **3.6 EVALUASI HASIL *CLUSTERING*.**

3.3 KONVERSI RUANG WARNA

Pada umumnya, pengolahan citra menggunakan model ruang warna RGB (*red, green, blue*) dalam merepresentasikan sebuah citra. *Red* adalah kanal merah, *green* adalah

kanal hijau, dan *blue* adalah kanal biru. Masing-masing kanal memiliki rentang nilai antara 0-255.

Adakalanya kita ingin menggunakan model ruang warna lain dalam mengolah suatu citra. Pada penelitian ini diterapkan dua ruang warna lain selain RGB, yaitu HSV dan L*a*b*.

1.3.1 Konversi Ruang Warna RGB ke HSV

Jika r, g, b ∈ [0,1] adalah koordinat merah, hijau, biru pada ruang warna RGB, dengan *max* adalah nilai terbesar dari r, g, b serta *min* adalah nilai terkecil dari r, g, b, maka untuk mencari derajat *hue* $h \in [0,360]$, *saturation*, dan *value* digunakan Persamaan 3.1.

$$
\begin{aligned}
h &= \begin{cases} 0 & \text{if } \max = \min \\ (60^\circ \times \frac{g-b}{\max - \min} + 360^\circ) \bmod 360^\circ, & \text{if } \max = r \\ 60^\circ \times \frac{b-r}{\max - \min} + 120^\circ, & \text{if } \max = g \\ 60^\circ \times \frac{r-g}{\max - \min} + 240^\circ, & \text{if } \max = b \end{cases} \\
s &= \begin{cases} 0, & \text{if } \max = 0 \\ \frac{\max - \min}{\max} = 1 - \frac{\min}{\max}, & \text{otherwise} \end{cases} \\
v &= \max
\end{aligned}
\tag{3.1}
$$

Pada Matlab, konversi ruang warna ini dilakukan dengan memanggil fungsi yang menggunakan potongan *code* berikut.

```
a1=rgb2hsv(a1); %konversi ruang warna RGB ke HSV
```

1.3.2 Konversi Ruang Warna RGB ke L*a*b*

Ruang warna L*a*b* berasal dari ruang warna XYZ dengan penambahan kanal pencahayaan (Wyszecki, 1982). Transformasi dari warna RGB ke XYZ dapat dilihat pada Persamaan 3.2.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.49 & 0.31 & 0.2 \\ 0.18 & 0.81 & 0.01 \\ 0 & 0.01 & 0.99 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{3.2}$$

Dan kemudian dari XYZ di transformasi menjadi L*a*b* dengan Persamaan 3.3.

$$\begin{aligned} L &= 116 \left(\frac{Y}{Y_n}\right)^{1/3} - 16 \\ a &= 500 \left[\left(\frac{X}{X_n}\right)^{1/3} - \left(\frac{Y}{Y_n}\right)^{1/3}\right] \\ b &= 200 \left[\left(\frac{Y}{Y_n}\right)^{1/3} - \left(\frac{Z}{Z_n}\right)^{1/3}\right] \end{aligned} \tag{3.3}$$

Pada aplikasi MATLAB, melakukan konversi ruang warna RGB ke L*a*b* dilakukan dengan potongan *code* berikut:

```
a=imread(D(i).name);%membaca nama citra
%membuat fungsi transformasi dari ruang warna RGB ke Lab
cform = makecform('srgb2lab');
%melakukan transformasi warna Lab terhadap setiap citra
hasil_lab = applycform(a,cform);
```

Pada akhir dari tahap ini dihasilkan sekumpulan piksel dalam bentuk matriks yang telah mengalami konversi ruang warna. Tahap selanjutnya adalah melakukan reduksi

matriks tersebut menjadi vektor (*feature extraction*) menggunakan *Gaussian Mixture Model (GMM)* dan histogram.

3.4 *GAUSSIAN MIXTURE MODEL* DAN HISTOGRAM

GMM pada tahap ini, *input* (masukan) nya adalah piksel dari citra (matriks), serta inisialisasi jumlah parameter *GMM* (jumlah *homogenous region* suatu citra). Sementara itu, *output* (luaran) dari *GMM* adalah probabilitas tiap *homogenous region* terhadap satu citra. *GMM* dilakukan pada tiap kanal warna pada setiap ruang warna yang digunakan, misalkan pada ruang warna RGB diterapkan *GMM* pada kanal warna merah, hijau, kemudian biru secara terpisah. Setelah itu, hasil *GMM* dari masing-masing kanal warna digabungkan dan kemudian disebut sebagai representasi dari citra tersebut (*feature space*) yang akan digunakan untuk dikelompokan.

Berikut ini potongan *code* iterasi *Expectation Maximization (EM)* untuk mendapatkan *maximum likelihood* yang merupakan parameter yang di butuhkan untuk menghasilkan *feature space* (hasil dari tahap *feature extraction*) yang berupa *prior probability* maksimal.

```
while(1)
   % Expectation
   prb = distribution(mu,sigma,p,ima);
   scal = sum(prb,2)+eps;
   loglik=sum(log(scal));%menghitung
                         loglikelihood
   %Maximization
   for j=1:k
     pp=prb(:,j)./scal;
     p(j) = sum(pp);
     mu(j) = sum(ima.*pp)/p(j);
     vr = (ima-mu(j));
     sigma(j)=sum(vr.*vr.*pp)/p(j)+sml;
   end
   p = p + 1e-3;
   p = p/sum(p);
```

```
    % Exit condition
    prb = distribution(mu,sigma,p,ima);
    scal = sum(prb,2)+eps;
    nloglik=sum(log(scal));
    if((nloglik-loglik)<1000) break; end;
end

function y=distribution(m,v,g,x)
x=x(:);
m=m(:);
v=v(:);
g=g(:);
for i=1:size(m,1)
    d = x-m(i);
    amp = g(i)/sqrt(2*pi*v(i));
    y(:,i) = amp*exp(-0.5 * (d.*d)/v(i));
end
%normalisasi terhadap NaN dan inf
ind=find(isnan(y)==1);
y(ind)=0;
ind=find(isinf(y)==1);
y(ind)=0;
```

Inisialisasi *mean*, varians dan *prior probability* yang digunakan dalam *GMM* dapat dilihat dalam potongan *code* berikut.

```
%inisialisasai parameter
mu=(1:k)*m/(k+1); %mean
sigma=ones(1,k)*m; %variance
p=ones(1,k)*1/k; %prior probability
```

Sementara itu, inisialisasi jumlah *homogenous region* pada setiap kanal warna adalah 10, sehingga total keseluruhannya ada 30 untuk masing-masing ruang warna yang digunakan (RGB, HSV, dan L*a*b*). Dengan demikian parameter yang ada pada sebuah citra yang berupa *prior probability*, *mean*, dan varians nya masing-masing berjumlah 30 untuk setiap ruang warna.

Kemudian, iterasi *EM* berhenti pada saat mencapai *threshold* 1000, yaitu saat nilai selisih logaritma *likelihood* yang baru dengan logaritma *likelihood* sebelumnya lebih

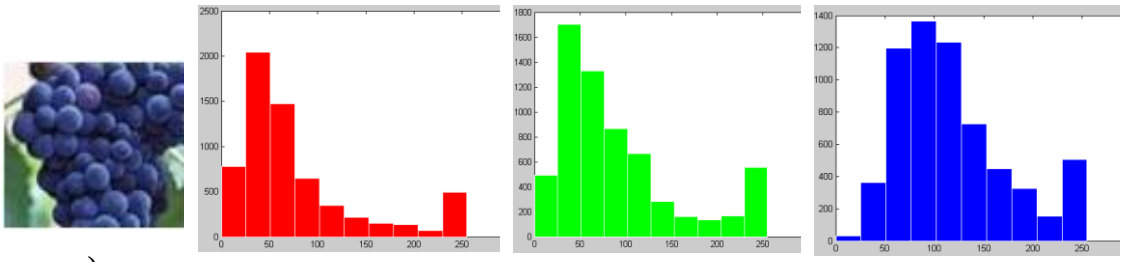
kecil dari 1000. *Threshold* ini dipilih karena menghasilkan kinerja yang lebih baik pada penelitian ini.

Prior probability dari setiap citra digunakan sebagai ciri dari citra hasil representasi menggunakan *GMM*. Parameter ini dipilih karena memiliki kinerja yang paling baik dibandingkan jika menggunakan *mean* atau varians nya. Jadi, luaran dari tiap citra setelah melalui tahap *feature extraction* adalah vektor yang terdiri atas 30 elemen yang merupakan ciri dari citra tersebut. Vektor ini digunakan sebagai masukan untuk melakukuan proses *clustering*.

Metode berikutnya adalah histogram yang dilakukan dengan menghitung distribusi jumlah piksel yang ada pada 10 interval pada tiap kanal warna, yaitu merah, hijau, dan biru untuk RGB, *hue, saturation,* dan *value* untuk HSV, serta pencahayaan, hijau-merah, dan biru-kuning untuk L*a*b*. Berikut ini adalah potongan *code* Matlab yang memanggil fungsi histogram.

```
a=rgb2hsv(a); %konversi ke hsv
% representasi histogram pada tiap komponen ruang warna
 for itr=1:3
   dlmwrite('hsv',a(:,:,itr));
   a1=dlmread('hsv',',');
   c=hist(a1(:));
   mtr=[mtr c];
end
```

Fungsi `hist` menghitung jumlah elemen dari vektor a1 yang berada pada setiap interval (terdiri atas 10 interval). Karena ada 3 kanal dalam tiap ruang warna (RGB, HSV, L*a*b*), maka jumlah elemen dalam vektor pada sebuah citra adalah 30. Untuk lebih jelasnya dapat dilihat pada Gambar 3.7.



a) b) c) d)

Gambar 3.7 Plot Histogram Buah Anggur pada Ruang Warna RGB

Gambar 3.7 a) adalah citra buah anggur, b) merupakan histogram kanal warna merah pada citra anggur, c) merupakan histogram kanal warna hijau, dan d) adalah histogram kanal warna biru.

Masing-masing kanal warna menghasilkan 10 elemen vektor, sehingga ketika digabungkan ketiga kanal tersebut dihasilkan vektor dengan 30 elemen yang kemudian digunakan sebagai masukan untuk tahap *clustering*.

3.5 PROSES *CLUSTERING*

Masing-masing vektor hasil dari histogram dan *GMM* citra di gabungkan ke dalam dua matriks berbeda yang siap di terapkan algoritma *clustering*. Penggabungan setiap vektor hasil representasi masing-masing citra dilakukan dengan cara penggabungan sebagai berikut:

```
mtrks=[mtrks;pb]; %menyimpan prior probabilitas sebagai hasil
representasi GMM ke dalam matriks
```

Setelah itu, dilakukan *clustering*, yang pertama dengan *Agglomerative Hierarchical Clustering*. Potongan *code* yang memanggil fungsi *AHC*:

```
idx=clusterdata(mtrks,5) %memanggil fungsi hierarchical
clustering
```

Metode *AHC* yang digunakan pada penelitian ini menggunakan skema *average link*. Alasan menggunakan skema *average link* adalah karena telah melakukan percobaan sebelumnya dan skema ini memiliki kinerja yang lebih baik daripada skema *single*, *complete*, ataupun *centroid link*. Adapun dalam penelitian yang pernah dilakukan pada *data clustering* (Adi, 2007) dihasilkan skema *average link* memiliki kinerja terbaik dibanding skema lainnya.

Hasil *clustering* yang menggunakan metode *AHC* tadi disalin ke dalam direktori yang sesuai dengan label indeks dari tiap *clustering* yang dihasilkan. Potongan *code* berikut adalah fungsi menyalin citra hasil *clustering* ke dalam direktori yang sesuai dengan label indeksnya. Misalnya, indeks hasil *clustering* berlabel 1 akan disalin ke dalam direktori berlabel 1.

```
%copy hasil clustering citra ke dalam direktori yang sesuai
dengan
%index clustering
for i=1:size(idx)
f1=fullfile('C:/Program Files/MATLAB71/work/TA/Lab/GMM/H/',
int2str(idx(i)));
    if (exist(f1) == 0)
        mkdir (f1);
    end
copyfile(nm(i,:),f1);
end
```

Berikut ini adalah potongan *code* yang memanggil fungsi *K-Means* yang ada pada Matlab.

```
%memanggil fungsi K-Means clustering
idx=kmeans(mtr,2,'EmptyAction', 'singleton')
```

Pengukuran jarak yang digunakan dalam *K-Means* adalah *euclidean distance*. Sama halnya dengan hasil *clustering AHC* yang disalin ke direktori, hasil *clustering K-Means* juga disalin ke direktori sesuai dengan indeks hasil *clustering*. Setelah itu, dilakukan evaluasi hasil *clustering* yang akan dibahas pada bagian berikutnya.

3.6 EVALUASI HASIL *CLUSTERING*

Setelah proses *clustering* berakhir, dilakukan evaluasi hasil *clustering* dengan mengevaluasi isi direktori yang menyimpan citra yang dikelompokan dan melakukan penghitungan jumlah citra yang berhasil masuk ke dalam kelompok yang tepat. Cara penghitungan jumlah citra tersebut adalah sebagai berikut.

1. Buka direktori yang berisi citra hasil *clustering* satu per satu.
2. *Rename* nama direktori sesuai dengan nama kategori dengan jumlah citra paling banyak (dominan) yang berada di direktori tersebut. Misalnya, untuk satu direktori yang didominasi oleh citra pinguin, maka ganti nama direktori tersebut menjadi pinguin.
3. Jika citra yang dominan pada satu direktori ternyata sudah menjadi nama dari direktori lain, maka pilih nama citra dominan kedua, dan seterusnya.
4. Lakukan langkah 1-3 sampai semua direktori telah diberi nama.
5. Hitung jumlah citra yang benar berada pada direktori yang sesuai dengan kategorinya.
6. Jumlah total dari citra yang masuk tepat pada direktori adalah jumlah citra yang dikatakan benar dalam proses *clustering*.