BAB 2
LANDASAN TEORI

Bab ini berisi penjelasan mengenai *image clustering,* pengukuran kemiripan dan pengukuran jarak, representasi citra, ruang warna, algoritma *clustering,* dan penelitian yang berhubungan.

2.1 IMAGE CLUSTERING

Clustering adalah suatu klasifikasi yang *unsupervised* (tidak terlatih) terhadap suatu *pattern* (data, *feature vector*) menjadi beberapa kelompok atau *cluster* berdasarkan kemiripannya (Jain, 1999). Secara intuitif, suatu *pattern* yang berbeda didalam suatu *cluster* yang benar akan lebih mirip satu sama lain jika dibandingkan dengan *pattern* yang berada pada *cluster* yang berbeda. Dapat juga diartikan sebagai proses untuk mendefinisikan pemetaan atau *mapping* $f:D \rightarrow C$ dari beberapa data $D=\{t_1, t_2, ..., t_n\}$ ke dalam beberapa *cluster* $C=\{c_1, c_2, ..., c_n\}$ berdasarkan kesamaan antar t_i. Sebuah *cluster* adalah sekumpulan objek yang digabung karena persamaan atau kedekatannya. Sedangkan *image clustering* adalah proses yang membagi atau mengelompokkan sekumpulan citra ke dalam beberapa bagian yang berbeda, dimana pada bagian yang berbeda tersebut anggotanya memiliki kesamaan khusus *(homogen).*

Terdapat perbedaan antara *clustering* dengan *discriminant analysis* (*supervised learning/categorization*). Didalam *supervised learning/categorization* (Jain, 1999), data merupakan sekumpulan *pattern* yang telah diberi label (telah diklasifikasikan terlebih dahulu), dan permasalahannya adalah memberikan label pada sebuah *pattern* baru yang belum berlabel. Umumnya *pattern* yang berlabel (*training pattern*)

dipergunakan untuk memberi label pada sebuah *pattern* baru. Sedangkan didalam *clustering*, permasalahannya adalah untuk mengelompokkan suatu *pattern* yang belum berlabel menjadi kumpulan *cluster* yang bermakna. Biasanya *cluster* yang diperoleh akan diasosiasikan menjadi label atau kelas. Label semacam ini dapat disebut sebagai *data driven* karena diperoleh secara langsung dari data.

Secara garis besar aktivitas *clustering* melibatkan beberapa tahapan (Jain, 1999), yaitu sebagai berikut.

1. Merepresentasikan *pattern* (dapat melibatkan *feature extraction* atau *feature selection*).
2. Mendefinisikan *pattern proximity* atau *similarity* yang sesuai dengan domain dari data.
3. Melakukan *clustering* sesuai dengan algoritma yang dipilih.
4. Membuat abstraksi data (jika diperlukan).
5. Mengevaluasi hasil *clustering.*

Tahapan pertama dalam *clustering* adalah merepresentasikan citra ke dalam suatu *pattern* atau model yang sesuai. Merepresentasikan suatu *pattern* merupakan salah satu pekerjaan yang cukup sulit karena dapat menyebabkan hasil *clustering* tidak optimal jika salah representasi *pattern*-nya. *Pattern Representation* adalah jumlah kelas, jumlah *pattern* yang tersedia dan jumlah, skala, tipe dari *feature* yang tersedia bagi algoritma *clustering* (Jain, 1999).

Penelitian ini melibatkan koleksi *pattern* citra yang akan di *cluster*. Bagaimana suatu *pattern* citra direpresentasikan akan dijelaskan pada sub bab **2.3 REPRESENTASI CITRA.**

Terdapat dua cara untuk mengurangi dimensi dari citra yang akan diolah, yaitu dengan melakukan *feature selection* atau *feature extraction. Feature selection* berarti mengidentifikasi subset dari *features* yang ada yang paling bermakna atau berguna untuk proses *clustering*. Sedangkan *feature extraction* adalah mempergunakan satu atau lebih informasi *input* citra untuk menghasilkan *feature-feature* baru yang akan dipergunakan dalam proses *clustering*. Tujuan yang akan diperoleh dengan melakukan *feature selection* atau *feature extraction* adalah untuk meningkatkan performa dari proses *clustering* dan meningkatkan efisiensi komputasi proses *clustering.*

Warna adalah komponen yang digunakan untuk merepresentasikan citra pada penelitian ini. Terdapat banyak cara dalam merepresentasikan warna pada sebuah citra dan dalam penelitian ini digunakan ruang warna RGB, HSV, dan L*a*b* yang dibahas lebih lanjut pada sub bab **2.4 RUANG WARNA.**

Tahapan kedua dalam melakukan *clustering* adalah menentukan *pattern proximity. Pattern proximity* biasanya diukur dengan sebuah fungsi jarak yang terdefinisikan untuk sepasang *pattern.* Suatu bentuk pengukuran jarak yang paling sederhana antara sepasang *pattern* adalah *euclidean* yang dapat juga digunakan untuk mengukur *dissimilarity* antara *pattern.* Selain itu terdapat pengukuran yang berdasarkan kemiripan antar *pattern* atau *similarity measurement*. Proses *clustering* mempergunakan informasi jarak antar *pattern* atau *similarity* antar *pattern* untuk menggabungkan *pattern* yang mirip satu sama lain menjadi *cluster.* Proses pengukuran ini hanya dilakukan pada metode *agglomerative hierarchical*, tidak digunakan pada metode *K-Means clustering*. Pengukuran jarak dan kemiripan secara detail dibahas pada sub bab **2.2 PENGUKURAN KEMIRIPAN DAN PENGUKURAN JARAK.**

Tahapan ketiga dalam melakukan *clustering* adalah melakukan pemilihan algoritma yang sesuai dan kemudian menggunakan algoritma tersebut untuk menghasilkan *cluster* citra. Untuk pembahasan mengenai algoritma *clustering* yang digunakan dalam tugas akhir ini yaitu *Agglomerative Hierarchical* dan *K-Means* dapat dilihat pada sub bab **2.5 ALGORITMA *CLUSTERING*.**

Tahapan keempat dalam melakukan *clustering* (jika dibutuhkan) adalah melakukan abstraksi data. Dalam konteks *clustering* (Jain, 1999), abstraksi data adalah menentukan deskripsi dari *cluster-cluster* yang dihasilkan biasanya berupa *pattern* yang representatif seperti *centroid* dari suatu *cluster*. Dalam tugas akhir ini tidak menggunakan tahapan abstraksi data karena tahapan ini dibutuhkan jika *image clustering* yang dilakukan akan dipergunakan untuk CBIR (*content based image retrieval*).

Tahapan kelima dalam melakukan *clustering* adalah melakukan evaluasi hasil *clustering* yang dihasilkan. Tahapan ini dilakukan dengan membandingkan hasil *clustering* dengan hasil dari beberapa metode yang digunakan dalam penelitian ini. Tiga hal utama yang menjadi perbandingan dalam penelitian ini adalah ruang warna, representasi citra, dan metode *clustering* yang digunakan. Pembahasan mengenai evaluasi hasil *clustering* ini akan dijelaskan lebih detail pada **BAB IV EKSPERIMEN DAN ANALISIS EKSPERIMEN.**

2.2 PENGUKURAN KEMIRIPAN DAN PENGUKURAN JARAK

Terdapat berbagai macam pengukuran kemiripan (*similarity*) atau jarak (*dissimilarity*) antar *pattern* yang dapat dipergunakan oleh berbagai macam algoritma *clustering.* Pengukuran kemiripan yang digunakan dapat mempengaruhi anggota dari

suatu *cluster*, selain itu tipe dari suatu *pattern*/data (kualitatif/kuantitatif) juga menentukan ukuran apa yang tepat untuk dipergunakan dalam suatu algoritma. Pengukuran kemiripan antar data dan pengukuran jarak antar data sepintas merupakan hal yang sama, namun sesungguhnya kedua hal itu berbanding terbalik, yaitu jika antar data kemiripannya sangat tinggi (nilai pengukuran tinggi), maka jarak antar data tersebut sangat dekat (nilai pengukuran rendah).

Untuk suatu himpunan data matriks *X* berukuran *m* x *n* yang beranggotakan $x_i \in X$, *i=1,2,3,..., n*. Tiap data x_i merupakan *feature vector* $x_i = \{x_{i1}, x_{i2}, ..., x_{im}\}$ dengan *m* adalah dimensi dari *feature vector* tersebut. Berikut ini adalah rumus-rumus yang dapat digunakan untuk mengukur jarak dan kemiripan antar data x_r dengan data x_s :

1. Euclidean Distance

Ukuran ini sering dipergunakan dalam *clustering* karena sederhana walaupun sangat sensitif terhadap pencilan. Ukuran ini memiliki masalah jika skala nilai atribut yang satu sangat besar dibandingkan nilai atribut lainnya. Oleh sebab itu, nilai-nilai atribut di normalisasi sehingga berada dalam kisaran 0 dan 1. Persamaan *euclidean distance* dapat dilihat pada Persamaan 2.1 (Adi, 2007).

$$d_{rs}^2 = (x_r - x_s)(x_r - x_s)' \qquad (2.1)$$

2. Mahalanobis Distance

Pengukuran ini dilakukan dengan menggunakan matriks varians dari data matriks. Pengukuran ini dapat dilakukan dengan menggunakan Persamaan 2.2.

$$d_{rs}^2 = (x_r - x_s)V^{-1}(x_r - x_s)' \quad (2.2)$$

Dimana *V* adalah matriks *covariance*. Jika *V* adalah matriks identitas, maka sama dengan Euclidean.

3. Minkowski Metric

Pengukuran ini merupakan bentuk umum dari rumus *Euclidean Distance* jika nilai p=2 (Persamaan 2.3). Dengan pengukuran ini, lebih banyak nilai numerik yang dapat ditempatkan pada jarak terjauh diantara dua vektor. Seperti pada *Euclidean Distance*, pengukuran ini juga memiliki masalah jika salah satu atribut dalam vektor memiliki rentang yang lebih lebar dibandingkan atribut-atribut lainnya.

$$d_{rs} = \left\{ \sum_{j=1}^{n} |x_{rj} - x_{sj}|^p \right\}^{\frac{1}{p}} \quad (2.3)$$

4. Cossine Similarity

Pengukuran ini cocok untuk kemiripan antar data yang dimensinya cukup tinggi. *Sim*(d_i,d_j) yang mengikuti model *Cossine* sering juga dinotasikan cosine(d_i,d_j) atau cos(d_i,d_j). Pengukuran ini dirumuskan pada Persamaan 2.4.

$$d_{rs} = \left(1 - x_r x'_s / (x'_r x_r)^{\frac{1}{2}} (x'_s x_s)^{\frac{1}{2}}\right) \quad (2.4)$$

Dalam penelitian ini digunakan pengukuran kemiripan *euclidean distance* pada metode *Agglomerative Hierarchical Clustering (AHC)* dan *K-Means*.

2.3 REPRESENTASI CITRA

Terdapat beragam cara yang dapat dipergunakan untuk merepresentasikan suatu citra atau sering dikenal dengan istilah *image feature extraction*. *Feature extraction* adalah suatu bentuk mereduksi dimensi data input. Tahapan ini diperlukan jika data input yang akan diproses adalah data yang sangat besar. *Feature extraction* merupakan representasi yang lebih sederhana dari sebuah data input yang sangat besar. Hal ini bertujuan untuk memperkecil *cost* komputasi.

Hingga saat ini terdapat banyak cara untuk melakukan *feature extraction* yang didasarkan komponen warna, tekstur, garis tepi, bentuk, dan lainnya, dapat juga berupa gabungan beberapa komponen. Algoritma yang digunakan dalam melakukan *feature extraction* antara lain *harmonic mean*, histogram, *Gaussian Mixture Model*, dan lain-lain.

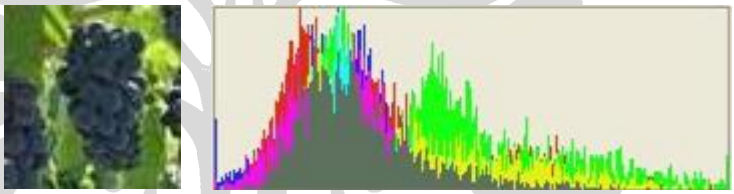
2.3.1 Histogram

Histogram merupakan representasi dari distribusi warna pada sebuah citra yang diperoleh dari menghitung jumlah piksel pada setiap citra. Citra histogram menunjukan probabilitas *mass function* pada citra. Bentuk formal dari histogram dapat dilihat pada Persamaan 2.5.

$$h_{A,B,C} = N \times Prob(A=a,B=b,C=c) \quad (2.5)$$

dimana A, B, dan C merepresentasikan 3 kanal warna (dalam RGB, HSV atau L*a*b*) dan N adalah jumlah piksel pada citra. Secara komputerisasi, histogram dibentuk dari *discretizing* warna pada sebuah citra dan menghitung jumlah piksel pada setiap warna. Histogram terdiri dari sekumpulan vektor. Untuk citra *gray-scale*, histogram berbentuk 2-dimensi vektor, satu dimensi untuk nilai *gray-level* dan yang satunya lagi menghitung jumlah piksel pada *gray-level*. Untuk citra warna, histogram berbentuk 4-dimensional vektor, yaitu 3 kanal warna dan satu untuk menghitung jumlah piksel pada suatu kanal. Keterbatasan dari histogram adalah kompleksitasnya yang ekponensial karena dimensi dari *feature space* nya.

Gambar 2.1 adalah salah satu contoh representasi histogram yang dilakukan pada citra anggur.



a) b)

Gambar 2.1 Plot Histogram Buah Anggur

Gambar 2.1 a) adalah citra buah anggur, b) merupakan histogram warna pada citra buah anggur.

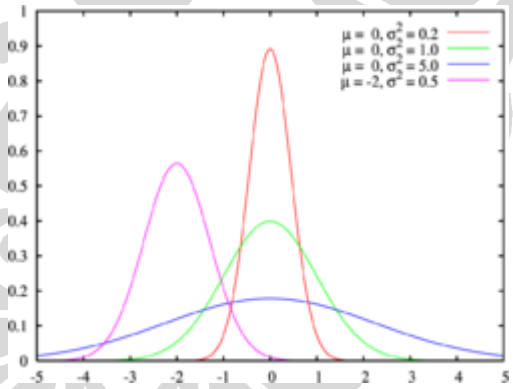
2.3.2 Gaussian Mixture Model

Sebuah percobaan dilakukan untuk mencari takaran optimal untuk membuat jus yang terdiri dari buah anggur, mangga, dan nanas. Diperlukan *mixture model* pada percobaan tersebut karena rasa yang dihasilkan bergantung pada jumlah takaran yang digunakan dari masing-masing buah. Contoh lain adalah keadaan finansial yang berubah dari waktu ke waktu yang terdiri dari situasi normal dan krisis. Sebuah

mixture model digunakan untuk menghasilkan data yang menggambarkan perbedaan pada perubahan keadaan itu.

Terdapat dua jenis *mixture model*, yaitu *direct* dan *indirect*. Keadaan finansial yang berubah-ubah pada contoh diatas adalah contoh aplikasi *direct* dari *mixture model*, sebuah keadaan dimana setiap observasi memberikan suatu komponen atau kategori yang berbeda. Dalam bentuk *mixture*, setiap komponen digambarkan oleh sebuah *probability density function* dan nilai *mixture* yang adalah probabilitas dari setiap komponen (Wikipedia, 2009).

Pada aplikasi *indirect, mixture model* digunakan untuk fleksibilitas matematis. Contohnya, sebuah *mixture* dari dua distribusi normal dengan *mean* dan *variance* yang berbeda pada sebuah populasi. Gambar 2.2 menunjukan contoh *mixture* dari empat distribusi normal atau distribusi *Gaussian*.



Gambar 2.2 *Mixture* dari Distribusi Normal

Terdapat tiga tipe dari *mixture model*, yaitu *probability mixture model*, *parametric mixture model*, dan *continuous mixture* (Wikipedia, 2009).

- *Probability Mixture Model*

  Model ini adalah sebuah distribusi probabilitas yang merupakan kombinasi konveks dari distribusi probabilitas lain. Misalkan variabel acak diskrit *X* adalah sebuah *mixture* dari *n* komponen variabel acak diskrit Y_i, maka fungsi mass probabilitas (*probability mass function*) *X*, $f_X(x)$ didefinisikan sebagai jumlah dari nilai distrbusi komponennya. Untuk $0 \leq a_i \leq 1$ dimana $a_1 + a_2 + ... + a_n = 1$, *d*efinisi tersebut juga digunakan untuk variabel acak kontinu *(continuous random variable)* dengan fungsi *f* adalah *probability density function* yang dapat dilihat pada Persamaan 2.6.

$$f_X(x) = \sum_{i=1}^{n} a_i f_{Y_i}(x) \tag{2.6}$$

- *Parametric Mixture Model*

  Pada model ini, distribusi komponen diperoleh dari Persamaan 2.7 dengan θ_i adalah parameter yang dicari.

$$f_X(x) = \sum_{i=1}^{n} a_i f_Y(x; \theta_i). \tag{2.7}$$

  Model inilah yang digunakan dalam penelitian ini karena pada penelitian ini memerlukan parameter *mean*, varians, dan probabilitas dari masing-masing matriks citra.

- *Continuous Mixture*

  Continuous mixture didefinisikan pada Persamaan 2.8.

$$f_X(x) = \int_\Theta h(\theta) f_Y(x;\theta)\, d\theta \tag{2.8}$$

dimana

$$\int_\Theta h(\theta)\, d\theta = 1. \text{ dan } 0 \leq h(\theta) \quad \forall\theta \in \Theta$$

Fungsi *Gaussian* pada umumnya, secara matematis dapat dituliskan pada Persamaan 2.9.

$$f(x) = ae^{-\frac{(x-b)^2}{2c^2}} \tag{2.9}$$

Jika $a=1/(\delta*(2\pi)^{1/2})$, $b=\mu$, dan $c=\delta$, dimana δ adalah varians dan μ adalah mean, maka dapat dituliskan seperti pada Persamaan 2.10.

$$f(x) = 1/(\delta*(2\pi)^{1/2})*\exp(-(x-\mu)^2/(2\delta^2)) \tag{2.10}$$

Persamaan 2.10 merupakan fungsi yang digunakan dalam *Gaussian Mixture Model* untuk melakukan estimasi terhadap nilai *posterior probability* dari setiap nilai piksel pada matriks citra dengan menggunakan *mean*, varians, dan *prior probability*. *Prior probability* adalah probabilitas dari satu *homogenous region* terhadap citra tersebut, sedangkan *posterior probability* adalah kemungkinan setiap piksel yang ada pada citra untuk masuk dalam setiap *homogenous region* yang ada pada satu citra. Akan dijelaskan lebih lanjut pada bagian tentang algoritma *Expectation-Maximization*.

Gaussian Mixture Model (GMM) adalah *mixture model* yang menggunakan distribusi *Gaussian*. Misalkan diberikan data masukan *X* dan jumlah *mixture M*, kemudian data tersebut disusun sebaik mungkin menggunakan distribusi *Gaussian* dari *M mixture*

(Gordon, 2006). Distribusi *Gaussian* diperoleh dari Persamaan 2.11 dimana α_j>0 dan $\sum_{j=1}^{k} \alpha_j = 1$, μ_j adalah *mean* (rata-rata), $\sum_j$ adalah *covariance* dan *y* adalah citra (matriks).

$$f(y) = \sum_{j=1}^{k} \alpha_j \frac{1}{\sqrt{(2\pi)^d |\Sigma_j|}} \exp\{-\frac{1}{2}(y-\mu_j)^T \Sigma_j^{-1}(y-\mu_j)\} \quad (2.11)$$

GMM menggunakan algoritma *Expectation Maximization (EM).* Sebenarnya, *GMM* yang digunakan pada proses *feature extraction* citra tidak melakukan *assign* piksel ke dalam *cluster*, tetapi keanggotaan dari setiap piksel pada sebuah *cluster* yang didefinisikan oleh sebuah *posterior probability*.

Setiap piksel memiliki nilai *posterior probability* sebanyak jumlah *cluster* yang ada dan total *probability* dari tiap piksel adalah sama. Sedangkan *cluster*-nya didefinisikan oleh sebuah *prior probability*, sebuah *cluster center* (*mean*), dan sebuah matriks *covariance*. *Center* dan *covariance* matriks menentukan *Mahalanobis distance* antara sebuah *cluster center* dan sebuah piksel. Untuk setiap *cluster* didefinisikan sebuah fungsi *Gaussian* dari *Mahalanobis distance* antara *cluster center* dengan piksel yang disebut sebagai sebuah fungsi *likelihood*.

Posterior probability pada *cluster* serta *center cluster (mean)* dan matriks *covariance* dihitung secara iteratif sampai menuju suatu nilai yang konvergen dengan menggunakan algoritme *EM*. Pada tahap *E*, setiap *cluster*, dihitung *posterior probability* nya. Sedangkan pada tahap *M*, semua *cluster center, prior probability*, dan matriks *covariance* dihitung menggunakan *posterior probability*. Proses ini berhenti pada saat memenuhi *stopping criterion*, dalam hal ini *threshold* sehingga

fungsi *likelihood* bernilai maksimum (Gordon, 2006). Secara matematis, algoritma *EM* ditulis sebagai berikut:

1. *Expectation* (Persamaan 2.12)

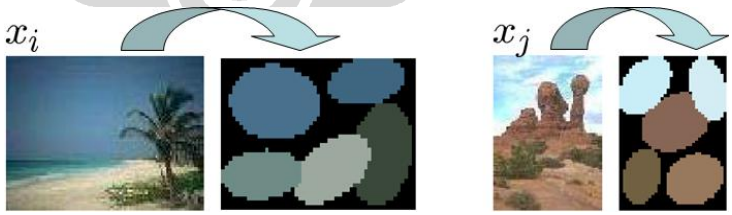
$$w_{tj} = \frac{\alpha_j f(y_t|\mu_j, \Sigma_j)}{\sum_{i=1}^{k} \alpha_i f(y_t|\mu_i, \Sigma_i)} \qquad j = 1, \ldots, k \quad , \quad t = 1, \ldots, n \tag{2.12}$$

2. *Maximization* (Persamaan 2.13)

$$\begin{aligned} \hat{\alpha}_j &\leftarrow \frac{1}{n} \sum_{t=1}^{n} w_{tj} \\ \hat{\mu}_j &\leftarrow \frac{\sum_{t=1}^{n} w_{tj} y_t}{\sum_{t=1}^{n} w_{tj}} \\ \hat{\Sigma}_j &\leftarrow \frac{\sum_{t=1}^{n} w_{tj} (y_t - \hat{\mu}_j)(y_t - \hat{\mu}_j)^T}{\sum_{t=1}^{n} w_{tj}} \end{aligned} \tag{2.13}$$

Akhir dari iterasi *EM* menghasilkan suatu nilai logaritma *likelihood* maksimum yang diperoleh dari $\sum log(f\ (Y_t|\mu_j, \sum_j))$. Kemudian, iterasi akan berhenti apabila selisih antara nilai logaritma *likelihood* yang baru dengan logaritma *ikelihood* sebelumnya lebih kecil dari *threshold* yang ditentukan.

Gambar 2.3 merupakan contoh hasil GMM dari citra RGB dimana masing-masing *homogenous region* yang terlihat pada bagian kanan memiliki *mean*, varians, dan *prior probability* masing-masing.



Gambar 2.3 Contoh Representasi GMM

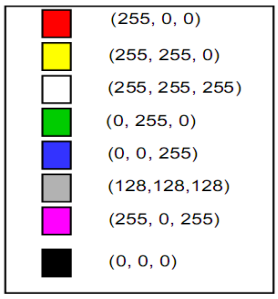
Hasil dari *Gaussian Mixture Model (GMM)* yang digunakan sebagai *feature extraction* pada penelitian ini adalah *prior probability* yang memiliki nilai *likelihood* maksimum. Inilah yang digunakan sebagai masukan untuk proses *clustering.*

2.4 RUANG WARNA

Terdapat berbagai macam ruang warna dalam komputerisasi, antara lain RGB, IHS, HSV, L*a*b*, L*u*v*, dan lain sebagainya. Pada bagian ini akan di jelaskan mengenai tiga ruang warna yang kemudian digunakan didalam penelitian ini, yaitu RGB, HSV, dan L*a*b*.

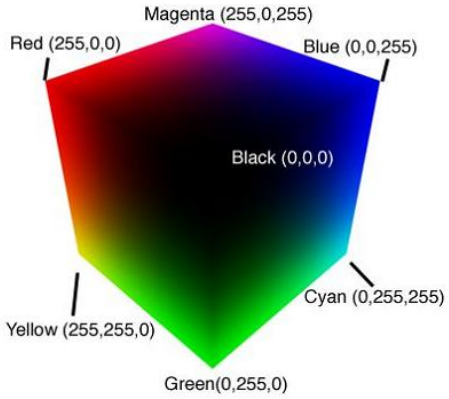
2.4.1 RGB

RGB adalah warna citra dijital yang umum digunakan karena merupakan warna pada tabung *display* dari monitor. RGB (*Red Green Blue*) yang disebut juga *true color* adalah representasi warna yang disimpan dalam M x N x 3 yang mendefinisikan kanal warna, yaitu merah, hijau, dan biru untuk setiap piksel. Jika *true color* itu digabungkan, maka akan menghasilkan warna lain. Penggabungan tersebut bergantung pada nilai *true color*, dimana tiap warna memiliki nilai 256 (8 *bit* atau 1 *byte*). Gambar 2.4 menunjukan konsep warna RGB. Warna yang dideskripsikan dalam RGB adalah pemetaan yang mengacu pada panjang gelombang dari RGB. Pemetaan menghasilkan nuansa warna untuk masing-masing R, G, dan B yang nilai diskritnya adalah 256 dengan indeks 0-255 (Wikipedia, 2009).



Gambar 2.4 Konsep Warna RGB

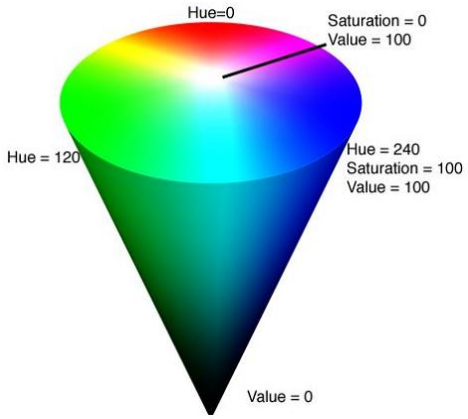
Dengan 3 *byte* warna ini, komputer dapat men-*generate* sebanyak 256 x 256 x 256 = 16.777.216 kombinasi warna. Misalkan warna merah adalah sumbu Z, warna hijau adalah sumbu Y, dan warna biru adalah sumbu X, maka dapat membentuk kubus warna RGB yang ditunjukan pada Gambar 2.5.



Gambar 2.5 Visualisasi ruang warna RGB

2.4.2 HSV

HSV merupakan singkatan dari *Hue*, *Saturation*, *Value*. *Hue* adalah paduan warna, *saturation* adalah kepekatan warna, dan *value* adalah nilai kecemerlangan warna (Cardani, 2006). Ruang warna ini memperhatikan unsur cahaya yang dapat digambarkan sebagai suatu bentuk kerucut seperti yang terlihat pada Gambar 2.6.



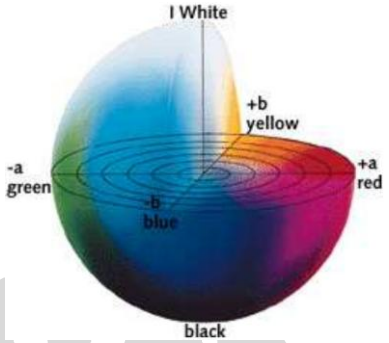
Gambar 2.6 Visualisasi ruang warna HSV

Pada Gambar 2.6, *hue* digambarkan sebagai lingkaran yang ada pada permukaan (kulit luar) kerucut dengan satuan derajat (^{o}) dimana 0^o berwarna merah, 120^o berwarna hijau, 240^o berwarna biru, dan seterusnya. *Saturation* dihitung dari titik tengah kerucut sampai pinggir kerucut yang memiliki rentang 0-100 dimana 0 berada di titik tengah kerucut dan 100 pada pinggiran kerucut. Sedangkan *value* dari bagian pangkal (permukaan lebar kerucut) yang bernilai 100 sampai ujung kerucut yang berbentuk runcing (bernilai 0) dimana bagian ujung berwarna hitam karena kecemerlangannya 0 dan permukaan lingkaran pada bagian pangkal yang berwarna putih karena kecemerlangannya maksimum, yaitu 100 (Cardani, 2006).

2.4.3 L*a*b*

Ruang warna *L*a*b** adalah interpretasi warna yang terdiri dari seberapa terang (L*), kecenderungan warna merah hijau (a*), dan kecenderungan warna biru kuning (b*) (Wikipedia, 2009). Seperti HSV, L*a*b* dirancang dengan mempertimbangkan penglihatan manusia yaitu unsur cahaya. L* memiliki rentang 0-100, dimana L*=0 menghasilkan hitam dan L*=100 menghasilkan warna putih. a* jika bernilai negatif

menghasilkan warna hijau, sedangkan positif menghasilkan warna merah. b* jika bernilai negatif menghasilkan biru, sedangkan positif menghasilkan kuning. Gambar 2.7 adalah visualisasi tiga dimensi dari ruang warna L*a*b* (Gordon, 2006).



Gambar 2.7 Visualisasi 3-D Ruang Warna Lab

2.5 ALGORITMA *CLUSTERING*

Terdapat berbagai macam teknik atau algoritma *clustering* yang dapat dipergunakan untuk mengelompokan data ke dalam *cluster* yang sesuai. Setiap teknik memiliki keunggulam dan kelemahan masing-masing. Bagian ini akan menjelaskan secara umum berbagai macam algoritma *clustering* untuk mengelompokkan citra ke dalam kelompok yang sesuai.

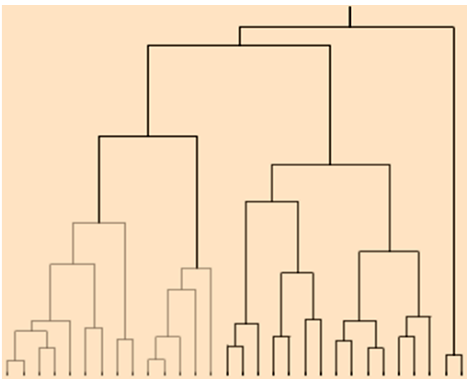
Ada beberapa isu yang berkaitan dengan teknik *clustering* (ADI, 2007), teknik *clustering* dapat berupa *agglomerative* atau dapat berupa *divisive.* Pendekatan secara *agglomerative* berarti proses *clustering* dimulai dengan setiap *pattern* merupakan satu kelompok, kemudian selama proses berlangsung, kelompok-kelompok yang mirip akan digabungkan, sampai terbentuk satu *cluster* besar yang tersusun oleh keseluruhan *pattern* atau sampai suatu *stopping criterion* tertentu terpenuhi.

Pendekatan secara *divisive* berarti proses *clustering* dimulai dengan keseluruhan *pattern* berada pada sebuah kelompok, kemudian melakukan *splitting* atau pemotongan sampai tersusun kelompok-kelompok yang hanya beranggotakan satu buah *pattern* atau sampai suatu *stopping criterion* tertentu terpenuhi.

Teknik *clustering* dapat berupa *hard* atau dapat berupa *fuzzy* (ADI, 2007). Suatu algoritma *clustering* yang *hard* berarti algoritma tersebut mengalokasikan setiap *pattern* hanya ke satu kelompok saja selama operasi berlangsung atau setelah hasil didapatkan. Suatu algoritma *clustering* yang *fuzzy* berarti algoritma tersebut memberikan sebuah nilai atau derajat keanggotaan bagi setiap *pattern* yang menjadi *input*-nya ke setiap kelompok yang dihasilkan. *Fuzzy clustering* dapat diubah menjadi *hard clustering* dengan cara memasukkan setiap *pattern* ke sebuah kelompok dengan nilai derajat keanggotaan tertinggi.

2.5.1 Algoritma *Hierarchical Clustering*

Algoritma *hierarchical clustering* menghasilkan *clustering* yang bertingkat dengan satu buah partisi yang tersusun atas keseluruhan data terletak diatas dan dengan kelompok-kelompok kecil yang tersusun atas satu data terletak dibawah. Masing-masing level perantara dapat dilihat sebagai kombinasi dua kelompok dari level terbawah sebelumnya atau melakukan pemotongan kelompok dari level teratas sebelumnya. Hasil dari algoritma *hierarchical* dapat dituangkan dalam bentuk *tree* yang disebut dengan *dendogram*. Gambar 2.8 menunjukan bentuk *dendogram*.



Gambar 2.8 Dendogram

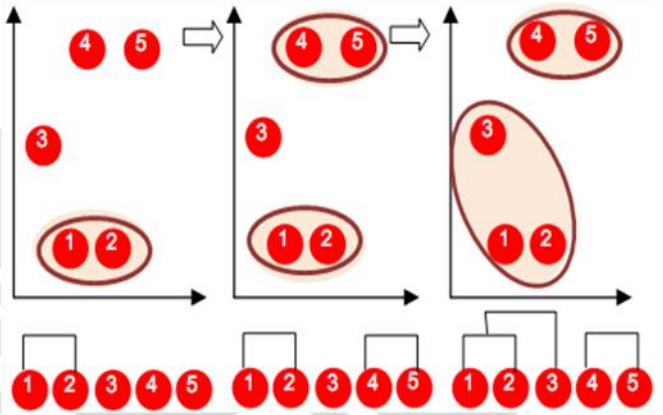
Ada dua pendekatan mendasar dalam menghasilkan *hierarchical clustering*, yaitu pendekatan *agglomerative* dan *divisive*. Pendekatan *agglomerative* dimulai dengan masing-masing data merupakan kelompok sendiri-sendiri kemudian dalam setiap langkah dilakukan penggabungan kelompok-kelompok yang paling mirip atau dekat jaraknya. Pendekatan *divisive* berarti berawal dari satu kelompok yang tersusun atas keseluruhan data, kemudian dalam setiap langkah dilakukan pemotongan sampai tersusun kelompok-kelompok yang hanya beranggotakan satu data. Dalam hal ini berarti dalam setiap langkah kita harus menentukan kelompok mana yang akan di potong dan bagaimana cara melakukan pemotongannya. Pendekatan secara *agglomerative* merupakan pendekatan yang paling umum dan yang akan dipergunakan dalam tugas akhir ini.

Berikut ini merupakan bentuk sederhana dari algoritma *hierarchical agglomerative clustering* (Adi, 2007):

Algoritma *Hierarchical Agglomerative Clustering*

1. Menentukan *k* sebagai jumlah *cluster* yang ingin dibentuk
2. Setiap data dianggap sebagai satu *cluster (trivial)*. Jika terdapat *N* jumlah data, dan *c* jumlah *cluster*, maka jumlah *c=N*

3. Menghitung jarak antar *cluster* dengan menggunakan salah satu persamaan jarak
4. Cari dua *cluster* yang memiliki jarak paling minimal dan gabungkan. Jumlah $c=c-1$.
5. Jika $c>k$, kembali ke-langkah 3.



Gambar 2.9 Ilustrasi Algoritma *AHC*

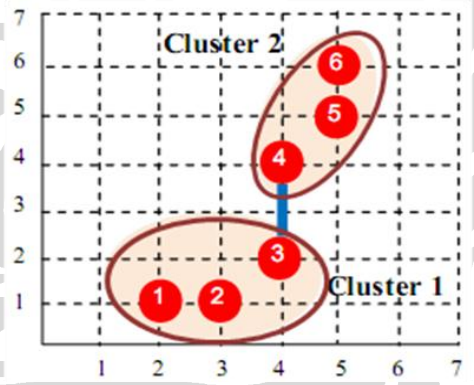
Gambar 2.9 menunjukan tahapan dari algoritma *AHC*. Parameter utama dalam algoritma *agglomerative hierarchical* adalah metode yang dipergunakan untuk menentukan pasangan kelompok yang akan digabungkan pada setiap langkah.

Terdapat beberapa skema yang dapat di gunakan dalam penghitungan jarak yang dilakukan pada *AHC*, yaitu *single link*, *complete link*, *centroid link*, dan *average link* (Lance, 1967).

a) Single link

Single link adalah proses *clustering* yang didasarkan pada jarak terdekat antar obyeknya (*minimum distance*) (Lance, 1967). Metode ini sangat baik digunakan untuk melakukan analisis pada setiap tahap pembentukan kelompok. Algoritma *single link* sebagai berikut.

1. Diasumsikan setiap data adalah satu kelompok data.
2. Menghitung jarak dua kelompok dengan menggunakan salah satu pengukuran jarak
3. Mencari dua kelompok yang mempunyai jarak antar kelompok paling minimal dan gabungkan
4. Kembali kelangkah 3 dan ulangi sampai mencapai kelompok data yang diinginkan



Gambar 2.10 Ilustrasi *Single Link*

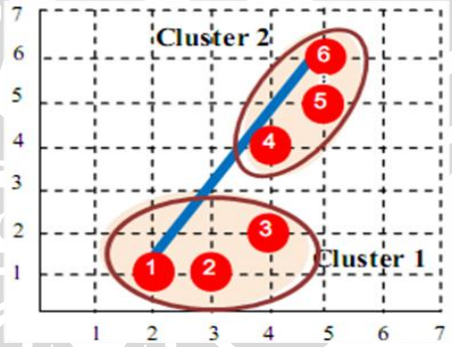
Pada Gambar 2.10, jarak kelompok 1 dengan kelompok 2 adalah jarak antara data 3 dan data 4.

b) Complete link

Complete link adalah proses *clustering* yang didasarkan pada jarak terjauh dari data antar kelompok *(maximum distance)* (Lance, 1967). Metode ini baik untuk kasus *clustering* dengan distribusi data normal. Akan tetapi tidak cocok untuk data yang mengandung *outlier* (pencilan).

Algoritma skema *complete link* adalah sebagai berikut.

1. Diasumsikan setiap data adalah satu kelompok data.
2. Menghitung jarak dua kelompok dengan menggunakan salah satu pengukuran jarak
3. Mencari 2 kelompok yang mempunyai jarak paling maksimal antar 2 kelompok dan gabungkan ke dalam satu kelompok baru
4. Kembali kelangkah 3 dan ulangi sampai mencapai kelompok data yang diinginkan



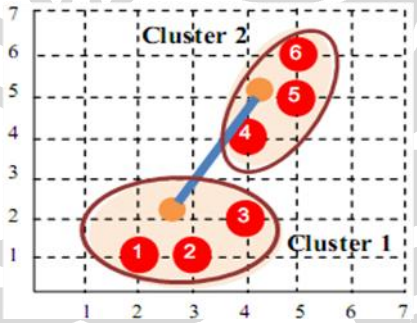
Gambar 2.11 Ilustrasi *Complete Link*

Pada Gambar 2.11, jarak kelompok 1 dengan kelompok 2 adalah jarak antara data 1 dan data 6.

c) *Centroid link*

Centroid link adalah proses *cluster* yang dilakukan berdasarkan jarak *centroid*-nya (Lance, 1967). Metode ini baik untuk data yang mengandung *outlier*. Algoritma *centroid link* sebagai berikut.

1. Diasumsikan setiap data adalah satu kelompok data.
2. Menghitung jarak dua kelompok dengan menggunakan salah satu pengukuran jarak
3. Mencari 2 kelompok yang mempunyai jarak antar *centroid* paling minimal antar 2 kelompok dan gabungkan ke dalam satu kelompok baru
4. Kembali kelangkah 3 dan ulangi sampai mencapai kelompok data yang diinginkan

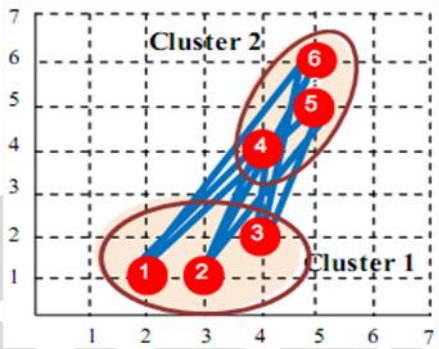


Gambar 2.12 Ilustrasi *Centroid Link*

Pada Gambar 2.12, jarak kelompok 1 dengan kelompok 2 adalah jarak antara *centroid* kelompok 1 dan *centroid* kelompok 2.

d) ***Average link***

Average link adalah proses *clustering* yang didasarkan pada jarak rata-rata antar obyek *(average distance)* (Lance, 1967). Skema ini merupakan skema yang relatif paling baik dari skema yang lain, namun waktu komputasinya lebih tinggi.



Gambar 2.13 Ilustrasi *Average Link*

Pada Gambar 2.13, jarak kelompok 1 dengan kelompok 2 adalah diperoleh dengan rumusan *(∑ jarak antar data)/(nxm)*. Dimana n adalah jumlah data kelompok 1 dan m adalah jumlah data kelompok 2.

2.5.2 Algoritma *Partitional Clustering*

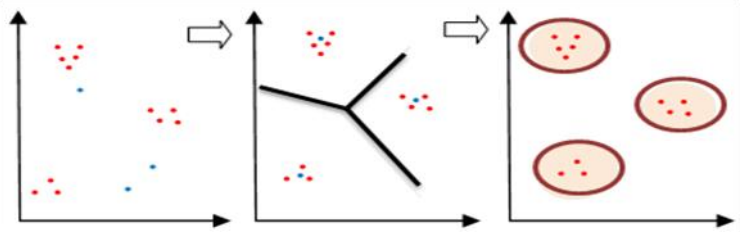
Algoritma *partitional clustering* menghasilkan sebuah solusi *clustering* yang tidak bertingkat atau *flat* karena tersusun atas satu level saja. Jika *k* merupakan jumlah kelompok yang diinginkan, maka pendekatan *partitional* umumnya akan mencari keseluruhan kelompok dalam sekali waktu. Berbeda dengan skema *hierarchical* yang tradisional dimana akan menggabungkan dua kelompok menjadi satu. Namun, pendekatan *hierarchical* pun dapat digunakan untuk menghasilkan partisi *k cluster* yang *flat* dan sebaliknya mengaplikasikan skema *partitional* secara berulang dapat

juga menghasilkan *hierarchical clustering.* Salah satu algoritma *partitional clustering* yang digunakan dalam tugas akhir ini adalah *K-Means*.

Ide dasar dari algoritma *K-Means* adalah bahwa *centroid* dapat merepresentasikan suatu *cluster* (Adi, 2007). *Centroid* dapat berupa nilai tengah atau nilai rata-rata dari suatu data. Algoritma ini diawali dengan pemilihan partisi inisial secara acak dan kemudian tetap memasukkan keseluruhan *pattern* (*reassignment*) ke kelompok yang sesuai berdasarkan *similarity* antara *pattern* dengan pusat dari satu kelompok atau *centroid* secara terus-menerus sampai semua kriteria konvergensi tertentu terpenuhi (misal, sampai tidak ada perubahan *centroid* dari keseluruhan kelompok atau sampai tidak ada lagi *reassignment* suatu *pattern* dari satu kelompok ke kelompok lainnya). Berikut ini adalah bentuk sederhana dari algoritma *K-Means* (Adi, 2007):

Algoritma *K-Means* untuk menghasilkan *k-cluster*

1. Pilih *k* buah data/*pattern* secara acak untuk merepresentasikan *k* inisial *centroid*
2. Masukkan setiap data/*pattern* ke *centroid* yang terdekat
3. Menghitung *centroid* baru dari suatu kelompok dengan menggunakan anggota kelompok saat ini
4. Ulangi langkah 2 dan 3 sampai posisi *centroid* baru dan *centroid* lama tidak sama.

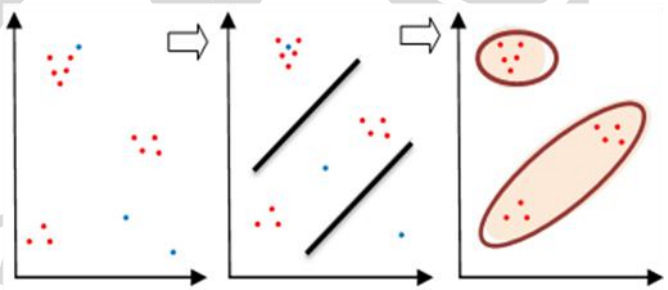


Gambar 2.13 Ilustrasi Algoritma *K-Means*

Gambar 2.13 menunjukan cara kerja algoritma *K-Means* pada sekumpulan data. Algoritma *K-Means* merupakan algortima yang cukup populer karena cukup mudah untuk diimplementasikan dan memiliki kompleksitas waktu yang linear atau *O(n)* dengan *n* adalah jumlah *pattern* atau data. Masalah yang umum dari algoritma ini adalah algoritma ini sangat sensitif terhadap pemilihan inisial partisinya dan menyebabkan *criterion function*-nya konvergen ke lokal minimum jika nilai dari inisial partisi tidak dipilih secara benar (Jain, 1999).

K-Means memiliki karakteristik sebagai berikut:

1. Sangat cepat dalam proses *clustering.*
2. Sangat sensitif terhadap pembangkitan *centroid* awal secara acak
3. Memungkin suatu kelompok tidak mempunyai anggota
4. Sangat sulit untuk mencapai global optimum.



Gambar 2.14 Ilustrasi Kelemahan Algoritma *K-Means*

Gambar 2.14 menunjukan salah satu kelemahan algoritma *K-Means* ketika ada pencilan.

2.6 PENELITIAN TERKAIT

Berikut ini adalah dua penelitian lain mengenai *image clustering* yang terkait dengan penelitian ini.

1. *Local vs Global Histogram–Based color Image Clustering*. Bongani Malinga, Daniela Raicu, Jacob Furst (Malinga, dkk, 2006).

Meneliti tentang dua teknik representasi citra yang menggunakan teknik histogram, yaitu berdasarkan informasi lokal histogram dan informasi global untuk mencari karakter citra. Hasil dari penelitian ini menyimpulkan bahwa informasi lokal histogram memiliki performa yang lebih baik dimana karakter citra lebih dekat hubungannya dengan citra yang dilihat oleh mata manusia. Algoritma *clustering* yang digunakan dalam penelitian ini adalah *K-Means.*

2. *Color and Texture-Based Image Segmentation Using EM and Its Application In Image Retrieval.* Serge Belongie, Chad Carson, Hayit Greenspan, Jitendra Malik (Belongie, dkk, 2007).

Menggunakan EM (*Expectation Maximization*) sebagai representasi citra yang melakukan kombinasi antara komponen warna dan tekstur yang koheren yang mendukung segmentasi citra. Disebut sebagai representasi *blobworld*. Aplikasi yang dihasilkan dapat digunakan *user* untuk melihat representasi sebuah citra yang di *submit*.