Jurnal Mahajana Informasi, Vo.4, No.2, 2019
e-ISSN:2527-8290

DETEKSI PENYAKIT LIMFOMA DENGAN MENGGUNAKAN ALGORITMA REGION GROWING

Allwin M. Simarmata, S.Kom, M.Kom[1)], Tondy Pranata Sinaga[2)], Rudianto[3)], Ondi William Simanjuntak[4)]
Program Studi Teknik Informatika Universitas Prima Indonesia
Jl. Sekip Medan Telp (061)-4578890
e-mail : pranatasinagatondy@gmail.com

Abstrak

Limfoma merupakan istilah umum untuk berbagai tipe kanker darah yang muncul dalam sistem limfatik, yang menyebabkan pembesaran kelenjar getah bening. Jumlah penderita limfoma saat ini dirasa cukup fantastis sehingga patut diwaspadai. Untuk itu, diharapkan hendaknya masyarakat lebih peduli terhadap deteksi dini kanker, khususnya limfoma, serta menambah pengetahuan mengenai penyakit limfoma agar jumlah penderita limfoma tidak semakin bertambah. Untuk membantu masyarakat dalam mendeteksi penyakit limfoma, maka perlu dirancang sebuah aplikasi pendeteksian penyakit kelenjar. Salah satu metode yang dapat digunakan adalah metode region growing. Dengan menerapkan algoritma region growing, maka dapat ditentukan dan dikenali stadium kanker limfoma berdasarkan pada gambar citra X-Ray yang dimasukkan. Hasil dari penelitian ini adalah aplikasi untuk mengenali dan mengidentifikasi jenis stadium kanker kelenjar getah bening (limfoma).

Kata Kunci: Limfoma, segmentasi citra, metode region growing, citra X-Ray

1. PENDAHULUAN

Seiring dengan semakin berkembangnya teknologi informasi dan komputer, maka semakin banyak penerapannya dalam kehidupan sehari-hari. Salah satu bidang ilmu yang banyak ditemukan penerapannya dalam kehidupan sehari-hari adalah bidang ilmu pengolahan citra. Pengolahan Citra Digital merupakan bidang ilmu informatika (komputer) yang mempelajari tentang bagaimana suatu citra itu dibentuk, diolah, dan dianalisis sehingga menghasilkan informasi yang dapat dipahami oleh manusia. Ilmu pengolahan citra ini banyak diterapkan dalam berbagai bidang, seperti di bidang medis atau kedokteran yaitu untuk mendiagnosa penyakit.

Limfoma merupakan istilah umum untuk berbagai tipe kanker darah yang muncul dalam sistem limfatik, yang menyebabkan pembesaran kelenjar getah bening. Menurut data GLOBOCAN (IARC) tahun 2012, limfoma merupakan salah satu dari sepuluh penyakit kanker terbanyak di dunia pada tahun 2012 (InfoDATIN, 2015). Berdasarkan hasil riset, diketahui bahwa prevalensi limfoma di Indonesia pada tahun 2013 adalah sebesar 0,06 $^{0}/_{00}$ atau diperkirakan sebanyak 14.905 orang (InfoDATIN, 2015). Jumlah penderita limfoma ini dirasa cukup fantastis sehingga patut diwaspadai. Untuk membantu masyarakat dalam mendeteksi penyakit limfoma, maka perlu dirancang sebuah aplikasi pendeteksian penyakit kelenjar.

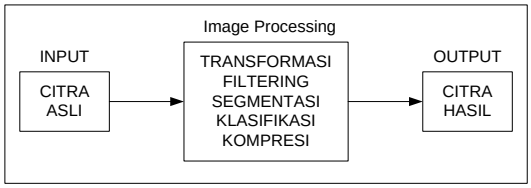
Untuk mendeteksi objek didalam citra maka perlu dilakukan proses segmentasi citra. Segmentasi citra bertujuan untuk membagi wilayah-wilayah yang homogen. Segmentasi adalah salah satu metode penting yang digunakan untuk mengubah citra input ke dalam citra output berdasarkan atribut yang diambil dari citra tersebut. Segmentasi membagi citra ke dalam daerah intensitasnya masing-masing sehingga bisa dibedakan objek-objeknya. Salah satu metode segmentasi yang dapat digunakan adalah metode *region growing*. *Region growing* merupakan sebuah prosedur yang mengelompokkan piksel-piksel atau subwilayah menjadi wilayah yang lebih besar berdasarkan kriteria yang sudah didefinisikan. Pendekatan dasarnya dimulai dari himpunan titik awal, kemudian wilayah diperbesar dengan menambahkan setiap titik piksel tetangga yang mempunyai sifat mirip dengan titik tersebut. Beberapa algoritma yang dapat digunakan untuk membagi suatu gambar (*image*) menjadi *region* adalah *Feature Selection Principal Component Analysis* (PCA), *Optimum Index Factory* (OIF) dan *Efficient Graph Based Algorithm*.

Algoritma berbasis *graph* yang efisien ini dikemukakan oleh Pedro F. Felzenszwalb dari Massachusetts Institute of Technology dan Daniel P. Huttenlocher dari Cornell University. Para ahli tersebut mendefinisikan suatu predikat untuk menghitung bukti dari sebuah batas antara dua daerah dengan menggunakan algoritma berbasis *graph*. Dengan menerapkan algoritma *region growing*, maka dapat ditentukan dan dikenali stadium kanker limfoma berdasarkan pada gambar citra X-Ray yang dimasukkan. Oleh karena itu, penulis memilih untuk membuat aplikasi identifikasi penyakit kelenjar yang mampu mengenali dan mengidentifikasi jenis stadium kanker dengan menggunakan metode *Region Growing*.

2. TINJAUAN PUSTAKA

2.1 Pengolahan Citra

Pengolahan Citra merupakan proses pengolahan dan analisis citra yang banyak melibatkan persepsi visual. Proses ini mempunyai ciri data masukan dan informasi keluaran yang berbentuk citra. Proses pengolahan citra (*Image Processing*) terdiri dari transformasi citra (*Image Transformation*), filtering, segmentasi citra (*Image Segmentation*), klasifikasi citra (*Image Classification*), dan kompresi citra (*Image Compression*). Proses pengolahan citra diperlihatkan pada Gambar 1.



Gambar 1. Proses Pengolahan Citra

Pengolahan citra digunakan terutama untuk memperjelas hasil *x-ray* organ tubuh manusia. Gambar yang didapat dari *x-ray* umumnya kabur sehingga sulit bagi para dokter untuk menganalisa kelainan-kelainan yang terdapat pada organ tubuh tersebut. Dengan pengolahan citra, gambar tersebut dapat diperjelas.

Transformasi citra adalah suatu proses perubahan dan perpindahan citra, yang meliputi proses perpindahan posisi citra, perubahan bentuk citra, ukuran citra dan perubahan isi citra. Transformasi citra ini terdiri dari transformasi dasar yang meliputi refleksi (pencerminan), penskalaan (*scalling*), rotasi (*rotation*) dan gusuran (*shearing*) serta transformasi lanjutan yang meliputi transformasi Fourier dan transformasi Wavelet.

Proses pengolahan citra umumnya menggunakan proses filtering untuk memperoleh hasil pengolahan yang diinginkan. Peningkatan mutu citra (*Image Enhancement*) merupakan pengolahan citra dengan menggunakan proses filtering untuk memberikan efek-efek tertentu pada suatu citra, seperti efek *transparant*, *comic*, *night*, *emboss* dan sebagainya. Proses pengolahan citra yang lain yang menggunakan proses filtering adalah perbaikan citra (*Image Restoration*) merupakan suatu proses yang dilakukan di mana suatu citra yang telah mengalami penurunan/degradasi dikembalikan ke bentuk citra aslinya. Penurunan tingkat/mutu citra dapat disebabkan oleh beberapa hal seperti penurunan tingkat kontras suatu citra yang menyebabkan citra tersebut sulit dibedakan, citra yang mengalami tingkat ketajaman sehingga menjadi lebih kabur dari citra aslinya, maupun citra yang mengalami kerusakan seperti rusak karena robekan, lipatan, timbulnya bintik-bintik/*spot* dan lain-lain.

Segmentasi citra adalah bagian dari teknik pengolahan citra. Segmentasi citra merupakan proses pengidentifikasian piksel-piksel ke dalam suatu kelas. Klasifikasi citra didefinisikan sebagai teknik ekstraksi dari kelas-kelas yang berbeda seperti pengelompokan daratan atau perairan yang datanya diperoleh dari satelit. Satu unit klasifikasi dapat berupa satu piksel, kumpulan piksel atau satu citra secara keseluruhan. Kompresi citra adalah pemadatan ukuran *file* citra, di mana citra tersebut disimpan dalam suatu format yang menggunakan formula matematika untuk menganalisis dan menyimpan data atau informasi citra yang didalamnya seperti panjang, lebar, kedalaman warna, atau informasi lainnya. Tujuan dari pemadatan data citra (kompresi citra) adalah mengurangi jumlah bit yang digunakan untuk menyimpan citra, dengan usaha semaksimum mungkin tidak terjadi penurunan mutu citra aslinya.

2.2 Computer Vision

Computer Vision merupakan proses analisis citra yang cirinya merupakan kebalikan dari Grafika Komputer. Data masukan biasanya merupakan suatu citra atau gambar dan proses yang dilakukan adalah proses pengalian struktur dengan hasil keluaran yang bersifat deskriptif. Sebagai contoh, pengenalan jenis penyakit paru-paru melalui citra sinar-x para penderita.

2.3 Segmentasi Citra

Segmentasi citra adalah pengidentifikasian piksel-piksel ke dalam suatu kelas. Umumnya, piksel dalam kelas yang sama serupa dengan yang lain, dan sebaliknya piksel dalam kelas yang berbeda tentunya saling berbeda pula. Dengan kata lain, segmentasi citra adalah membagi sebuah citra menjadi wilayah-wilayah yang homogen berdasarkan kemiripan tingkat keabuan suatu piksel dengan tingkat keabuan piksel-piksel sekitarnya. Proses segmentasi umumnya dilakukan terhadap citra *gray level*. Ada beberapa metoda dalam segmentasi citra yaitu *Fuzzy Clustering*, *Region Growing* dan *Edge Detection*. Metoda *Fuzzy Clustering* merupakan pendekatan *fuzzy* tertua dalam segmentasi citra. Algoritma seperti *Fuzzy C-Means* (FCM,Bezdek) dan *Possibilistic C-Means* (PCM, Krishnapuram & Keller) dapat digunakan untuk membangun cluster (segmen).

Keanggotaan dari piksel dapat ditafsirkan sebagai kemiripan dengan sebuah objek tertentu. Metoda *Region Growing* dilakukan dengan membangun wilayah berdasarkan piksel-piksel tunggal serta melakukan tes keseragaman terhadap kelompok-kelompok piksel tersebut di mana tes ini diterapkan pada masing-masing wilayah. Apabila ternyata wilayah tersebut tidak seragam, maka wilayah tersebut akan dipecah kembali menjadi wilayah-wilayah yang lebih kecil. Proses ini diulang sampai seluruh wilayah seragam. Metoda *Edge Detection* bertujuan untuk meningkatkan penampakan batas/pinggiran pada citra. Citra yang

diberikan akan dipartisi atau dibagi dalam segmen-segmen tertentu dengan mendeteksi batas-batas (*edges*) antara satu segmen dengan segmen lainnya dalam sebuah citra sehingga didapatkan batas-batas antara satu atau lebih objek dalam citra tersebut. Segmentasi citra dengan *Edge Detection* sering juga disebut sebagai proses penyederhanaan citra.

2.4 Region Growing

Region growing adalah sebuah metode segmentasi citra berbasis daerah yang sederhana. Metode ini juga diklasifikasikan sebagai sebuah metode segmentasi citra berbasis piksel karena metode ini mencakup proses seleksi dari titik-titik awal. Pendekatan segmentasi ini memeriksa piksel tetangga dari titik-titik awal dan menentukan apakah piksel tetangga harus digabung ke daerah (*region*). Proses ini bersifat iterasi (perulangan), dengan perilaku yang sama dengan algoritma pengelompokkan data (*data clustering*) biasa.

Sasaran utama dari segmentasi adalah untuk membagi sebuah citra menjadi daerah-daerah. Beberapa metode segmentasi seperti "*Thresholding*" mencapai sasaran ini dengan mencari batas antara daerah dengan berdasarkan pada ketidakkontinuan pada level keabuan atau properti warna. *Region-based segmentation* adalah sebuah teknik untuk menentukan daerah (*region*) secara langsung. Rumusan dasar untuk *region-based segmentation* adalah sebagai berikut:

$(a)\ \bigcup_{i=1}^{n} R_i = R.$

$(b) R_i$ is a connected region, $i = 1, 2, \ldots, n$

$(c)\ R_i \bigcap R_j = \varnothing$ for all $i = 1, 2, \ldots, n.$

$(d) P(R_i) = TRUE$ for $i = 1, 2, \ldots, n.$

$(e) P(R_i \bigcup R_j) = FALSE$ for any adjacent region R_i and R_j

P(Ri) adalah sebuah predikat logika yang didefiniskan melalui titik pada kumpulan P(Rk) dan $\varnothing$ adalah kumpulan kosong (*null set*).

(a) berarti segmentasi harus lengkap, yaitu setiap piksel harus berada pada daerah (*region*).
(b) memerlukan titik pada sebuah daerah harus dihubungkan berdasarkan definisi yang telah ditentukan sebelumnya.
(c) mengindikasikan bahwa daerah harus terpisah.
(d) berurusan dengan properti yang harus dipenuhi oleh piksel pada sebuah daerah tersegmentasi.
(e) mengindikasikan bahwa daerah R_i dan R_j adalah berbeda berdasarkan definisi dari predikat P.

2.5 Algoritma *Region Growing Efficient Graph-based*

Pedro F. Felzenszwalb dan Daniel P. Huttenlocher mengambil sebuah pendekatan berbasis *graph* untuk melakukan proses segmentasi. Anggap G = (V, E) adalah sebuah *graph* tak berarah dengan *vertex* $v_i \in V$, kumpulan dari elemen yang akan disegmentasi dan *edge* $(v_i, v_j) \in E$ yang berkorespondensi dengan pasangan dari *vertex* tetangga. Setiap *edge* $(v_i, v_j) \in E$ memiliki sebuah bobot berkorespondensi $w((v_i, v_j))$ yang merupakan sebuah ukuran non-negatif dari ketidaksamaan antara elemen tetangga v_i dan v_j. Pada kasus segmentasi citra, elemen pada V adalah piksel dan berat dari *edge* adalah ukuran tertentu dari ketidaksamaan antara dua piksel yang terhubung oleh *edge* tersebut (contoh: perbedaan pada intensitas, warna, gerakan, lokasi atau atribut lokal lainnya).

Pada pendekatan berbasis *graph*, sebuah segmentasi S adalah sebuah partisi dari V ke dalam komponen sedemikian sehingga setiap komponen (atau daerah) $C \in S$ berkorespondensi pada sebuah komponen terhubung pada sebuah *graph* G' = (V, E'), dimana $E' \geq E$. Dengan perkataan lain, sembarang segmentasi dikelompokkan oleh sebuah *subset* dari *edge* pada E. Terdapat beberapa cara untuk mengukur kualitas dari sebuah segmentasi, tetapi secara umum, penemu algoritma ini menginginkan agar elemen pada sebuah komponen harus mirip dan elemen pada komponen berbeda harus tidak mirip. Hal ini berarti bahwa *edge* antara dua *vertex* pada komponen yang sama harus memiliki bobot yang relatif rendah dan *edge* antara *vertex* pada komponen berbeda harus memiliki bobot yang lebih tinggi.

Dengan input berupa sebuah *graph* G = (V, E) dengan n *vertex* dan m *edge*. Output-nya adalah sebuah segmentasi dari V ke dalam komponen S = (C_1, …, C_r). Algoritma berbasis *graph* ini dapat dirincikan sebagai berikut:

1. Urutkan E ke dalam π = (o_1, …, o_m) berdasarkan bobot *edge* yang tidak berkurang.
2. Mulai dengan sebuah segmentasi S^0, dimana setiap *vertex* pada v_i adalah komponennya.
3. Ulangi langkah 4, untuk q = 1, ..., m.
4. Konstruksikan S^q dengan diberikan S^{q-1} sebagai berikut:
   Anggap v_i dan v_j melambangkan *vertex* yang terkoneksi oleh *edge* ke-q pada urutan, yaitu: $o_q = (v_i, v_j)$. Jika v_i dan v_j adalah komponen yang tidak terhubung dari S^{q-1} dan $w(o_q)$ bernilai kecil dibandingkan dengan perbedaan internal dari kedua komponen, maka gabungkan kedua komponen. Jika tidak, maka proses tidak dilakukan. Lebih formal lagi, anggap C_i^{q-1} adalah komponen dari S^{q-1} yang terdiri dari v_i dan C_j^{q-1} terdiri dari v_j. Jika $C_i^{q-1} \neq C_j^{q-1}$ dan $w(o_q) \geq \text{MInt}(C_i^{q-1}, C_j^{q-1})$ maka S^q telah diperoleh dari S^{q-1} dengan menggabungkan C_i^{q-1} dan C_j^{q-1}. Jika tidak, maka $S^q = S^{q-1}$.
5. Kembalikan nilai $S = S^m$.

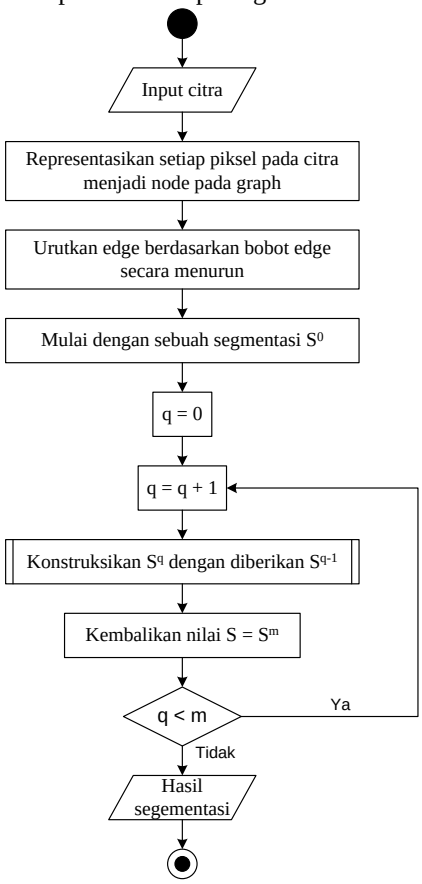
3. METODE PENELITIAN

Proses kerja sistem dimulai dari penginputan citra input yang akan disegmentasi. Setelah itu, setiap piksel pada citra direpresentasikan menjadi sebuah *node* pada *graph*, dengan penentuan sisi berdasarkan nilai piksel pada citra. Rumusan yang digunakan untuk menentukan nilai sisi yang menghubungkan *node* pada *graph* adalah sebagai berikut:

$$Sisi(i, j) = \sqrt{(R_i^2 + G_i^2 + B_i^2)} - \sqrt{(R_j^2 + G_j^2 + B_j^2)}$$

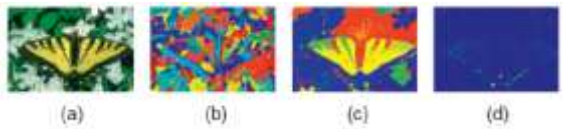
Setelah itu, maka setiap sisi pada *graph* akan diurutkan dengan urutan menurun. Proses segmentasi akan dimulai dengan mengganggap setiap piksel termasuk ke dalam sebuah *cluster*. Proses segmentasi akan dilakukan dengan menggabungkan setiap *node* pada *graph* yang memenuhi ketentuan dari algoritma. Proses segmentasi memanfaatkan algoritma Kruskal untuk mencari *minimum spanning tree* dari *graph* yang dibentuk. Hasil akhir dari proses segmentasi adalah kumpulan *cluster* yang mengelompokkan *piksel-piksel* pada citra.

Algoritma *Region Growing Efficient Graph-based* dapat dideskripsikan dalam bentuk *activity diagram* seperti terlihat pada gambar berikut:



Gambar 2. *Activity Diagram* dari Algoritma

Berikut diberikan contoh hasil segmentasi dengan menggunakan *efficient graph-based segmentation*:



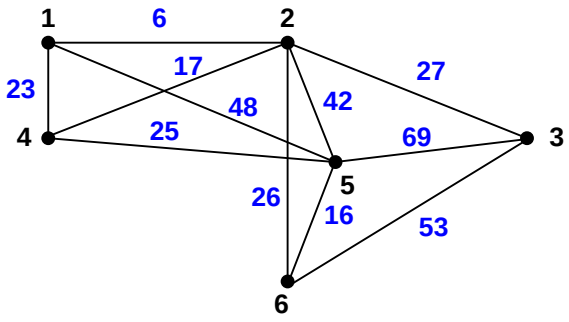
Gambar 3. Contoh Hasil Segmentasi dengan *Efficient Graph-based Segmentation*
(a) Citra Asli, (b), (c), dan (d) adalah hasil *efficient graph-based segmentation* dengan faktor normalisasi $h_s = 7$ dan faktor normalisasi warna $h_r = 7$ serta nilai k = 5, 25 dan 125.

Sebagai contoh, misalkan diketahui matriks warna dari sebuah citra sebagai berikut:

140	134	161
117	92	108

Selisih warna dari piksel pada sebuah citra direpresentasikan dalam bentuk *graph* seperti berikut:

- Sisi yang menghubungkan piksel 1 dan 2 berbobot 140 – 134 = 6
- Sisi yang menghubungkan piksel 1 dan 4 berbobot 140 – 117 = 23
- Sisi yang menghubungkan piksel 1 dan 5 berbobot 140 – 92 = 48
- Sisi yang menghubungkan piksel 2 dan 3 berbobot ABS(134 – 161) = 27
- Sisi yang menghubungkan piksel 2 dan 4 berbobot 134 – 117 = 17
- Sisi yang menghubungkan piksel 2 dan 5 berbobot 134 – 92 = 42
- Sisi yang menghubungkan piksel 2 dan 6 berbobot 134 – 108 = 26
- Sisi yang menghubungkan piksel 2 dan 5 berbobot 134 – 92 = 42
- Sisi yang menghubungkan piksel 3 dan 5 berbobot 161 – 92 = 69
- Sisi yang menghubungkan piksel 3 dan 6 berbobot 161 – 108 = 53
- Sisi yang menghubungkan piksel 4 dan 5 berbobot 117 – 92 = 25
- Sisi yang menghubungkan piksel 5 dan 6 berbobot ABS(92 – 108) = 16



Gambar 4. *Graph* G yang merupakan representasi dari piksel pada citra

Jumlah simpul dalam *graph input* G ada sebanyak n = 6 buah berarti jumlah langkah ada sebanyak n = 6 langkah. Langkah kerja dari

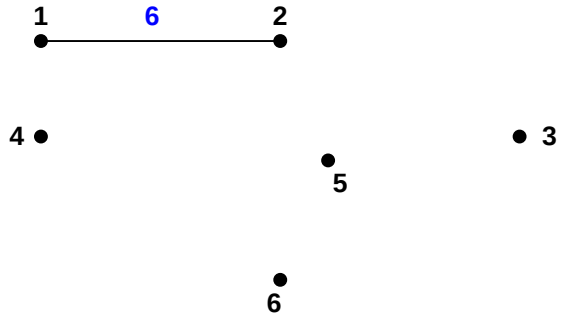
algoritma Kruskal tersebut dapat dirincikan sebagai berikut:

1. Urutkan semua sisi berdasarkan bobot (*weight*)-nya secara *ascending*.

Tabel 1. Urutan Sisi dalam *Graph Input* G secara *Ascending*

Sisi	Bobot
(1,2)	6
(5,6)	16
(2,4)	17
(1,4)	23
(4,5)	25
(2,6)	26
(2,3)	27
(2,5)	42
(1,5)	48
(3,6)	53
(3,5)	69

2. Ambil sebuah sisi yang berbobot paling minimum dari *graph input* G dan tidak membentuk sirkuit dalam T. Masukkan sisi tersebut ke dalam *graph* hasil T. Sisi yang diambil adalah sisi (1,2) yang berbobot 6. *Graph* hasil T masih kosong, maka setelah langkah (2) ini, *graph* hasil T hanya terdapat sisi (1,2) saja.



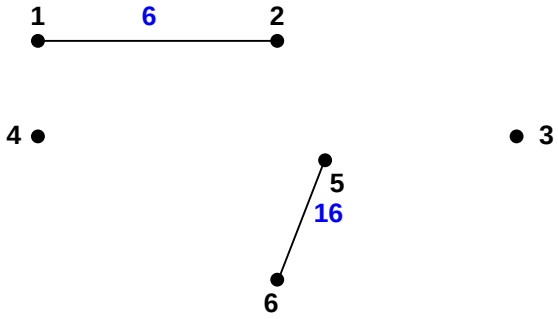
Gambar 5. Bentuk *Graph* Hasil T setelah langkah 2

Setelah langkah (2) ini, maka tabel kumpulan sisi pada *graph input* G yang belum dimasukkan ke dalam *graph* hasil T adalah:

Tabel 2. Tabel Kumpulan Sisi dalam *Graph Input* G yang belum dimasukkan ke dalam *Graph* Hasil T setelah langkah (2)

Sisi	Bobot
(5,6)	16
(2,4)	17
(1,4)	23
(4,5)	25
(2,6)	26
(2,3)	27
(2,5)	42
(1,5)	48
(3,6)	53
(3,5)	69

3. Ambil sebuah sisi yang berbobot paling minimum dari *graph input* G dan tidak membentuk sirkuit dalam T. Masukkan sisi tersebut ke dalam *graph* hasil T. Sisi yang diambil adalah sisi (5,6) yang berbobot 16. *Graph* hasil T telah terdapat sisi (1,2), maka setelah langkah (3) ini, *graph* hasil T terdapat sisi (1,2) dan (5,6).



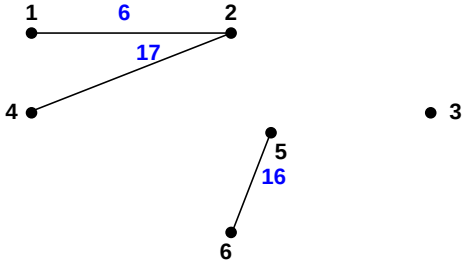
Gambar 6. Bentuk *Graph* Hasil T setelah langkah 3

Setelah langkah (3) ini, maka tabel kumpulan sisi pada *graph input* G yang belum dimasukkan ke dalam *graph* hasil T adalah:

Tabel 3. Tabel Kumpulan Sisi dalam *Graph Input* G yang belum dimasukkan ke dalam *Graph* Hasil T setelah langkah (3)

Sisi	Bobot
(2,4)	17
(1,4)	23
(4,5)	25
(2,6)	26
(2,3)	27
(2,5)	42
(1,5)	48
(3,6)	53
(3,5)	69

4. Ambil sebuah sisi yang berbobot paling minimum dari *graph input* G dan tidak membentuk sirkuit dalam T. Masukkan sisi tersebut ke dalam *graph* hasil T. Sisi yang diambil adalah sisi (2,4) yang berbobot 17. *Graph* hasil T telah terdapat sisi (1,2) dan (5,6), maka setelah langkah (4) ini, *graph* hasil T terdapat sisi (1,2), (5,6) dan (2,4).



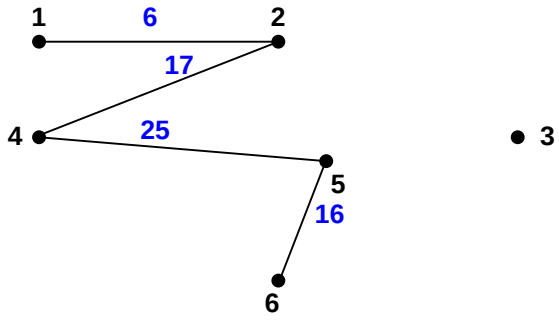
Gambar 7. Bentuk *Graph* Hasil T setelah langkah 4

Setelah langkah (4) ini, maka tabel kumpulan sisi pada *graph input* G yang belum dimasukkan ke dalam *graph* hasil T adalah:

Tabel 4. Tabel Kumpulan Sisi dalam *Graph Input* G yang belum dimasukkan ke dalam *Graph Hasil* T setelah langkah (4)

Sisi	Bobot
(1,4)	**23**
(4,5)	25
(2,6)	26
(2,3)	27
(2,5)	42
(1,5)	48
(3,6)	53
(3,5)	69

5. Ambil sebuah sisi yang berbobot paling minimum dari *graph input* G dan tidak membentuk sirkuit dalam T. Masukkan sisi tersebut ke dalam *graph* hasil T. Sisi (1,4) tidak diambil karena akan membentuk sirkuit pada *graph*. Sisi yang diambil adalah sisi (4,5). *Graph* hasil T telah terdapat sisi (1,2), (5,6) dan (2,4), maka setelah langkah (5) ini, *graph* hasil T terdapat sisi (1,2), (5,6), (2,4) dan (4,5).

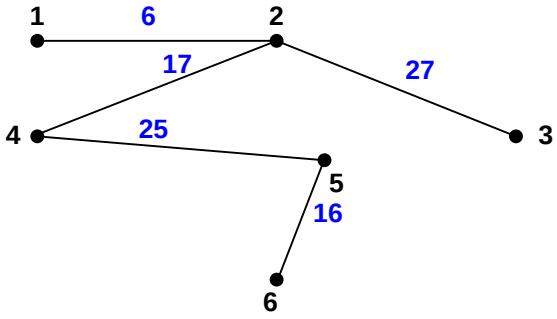


Gambar 8. Bentuk *Graph* Hasil T setelah langkah 5

Tabel 5. Tabel Kumpulan Sisi dalam *Graph Input* G yang belum dimasukkan ke dalam *Graph* Hasil T setelah langkah (5)

Sisi	Bobot
(2,6)	**26**
(2,3)	27
(2,5)	42
(1,5)	48
(3,6)	53
(3,5)	69

6. Ambil sebuah sisi yang berbobot paling minimum dari *graph input* G dan tidak membentuk sirkuit dalam T. Masukkan sisi tersebut ke dalam *graph* hasil T. Sisi (2,6) tidak dapat dipilih karena membentuk sirkuit. Sisi yang diambil adalah sisi (2,3) yang berbobot 27. *Graph* hasil T telah terdapat sisi (1,2), (5,6), (2,4) dan (4,5), maka setelah langkah (5) ini, *graph* hasil T terdapat sisi (1,2), (5,6), (2,4), (4,5) dan (2,3).



Gambar 9. *Minimal Spanning Tree* dari *Graph Input* G

Jumlah sisi yang terdapat dalam *graph* hasil T ada sebanyak 5 buah dan jumlah simpul ada sebanyak 6 buah, maka persyaratan bahwa

[jumlah sisi] = [jumlah simpul] – 1

telah terpenuhi, berarti proses telah selesai dan *graph* hasil T merupakan *minimal spanning tree* dari *graph input* G.

Misalkan pertama kali dipilih sisi e(1) = (1, 2) = 6 dengan xi = 1 dan xj = 2, maka berdasarkan hasil MST diatas, maka Li = 4 dan Lj = 3.

Misalkan nilai k = 5, maka:

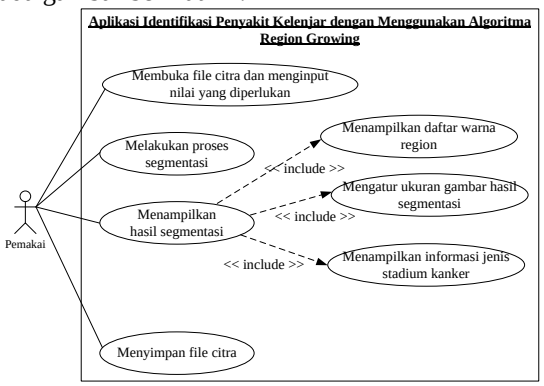
Temp1 = Li + k / C(0,1) = 4 + 5 / 1 = 9

Temp2 = Lj + k / C(0,2) = 3 + 5 / 1 = 8

Nilai minimumnya adalah Temp = Temp2 = 8

Karena e(1) = 6 lebih kecil daripada nilai Temp, maka C(0, 1) dan C(0, 2) digabungkan menjadi sebuah *cluster*.

Hubungan antara fungsi-fungsi diatas dapat digambarkan dalam bentuk diagram seperti terlihat pada gambar berikut ini:



Gambar 10. *Use Case* Sistem

4. HASIL DAN PEMBAHASAN

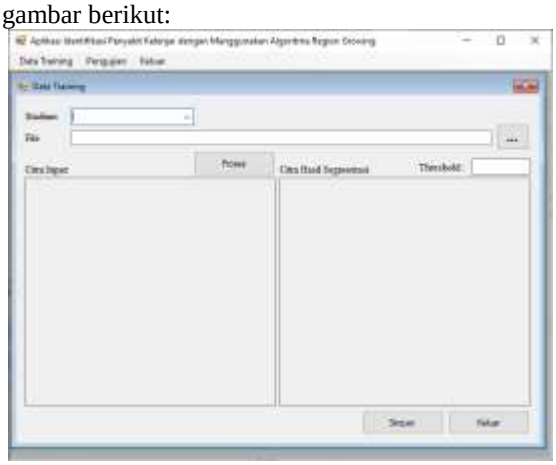
Apabila program dijalankan, maka tampilan yang pertama kali muncul adalah *form* ‘Awal’ seperti terlihat pada gambar 11 berikut.

Gambar 11. Tampilan *Form* 'Awal'

Di dalam *form* 'Awal' terdapat tiga buah *menu / link* utama yang berfungsi sebagai *link* ke *form-form* lainnya yang terdapat pada program. Adapun *link* yang terdapat dalam perangkat lunak adalah sebagai berikut:

1. *Link* 'Data Training', digunakan untuk memasukkan data citra yang akan digunakan sebagai *dataset*.
2. *Link* 'Pengujian', digunakan untuk melakukan proses identifikasi penyakit kelenjar dengan menggunakan metode *Region Growing*.
3. *Link* 'Keluar', berfungsi untuk menutup aplikasi.

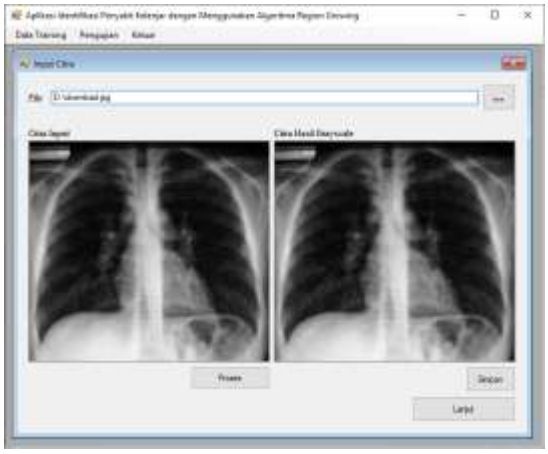
Untuk memasukkan data citra yang akan digunakan sebagai dataset, maka *user* dapat mengklik *link* 'Training', seperti terlihat pada gambar berikut:



Gambar 12. Tampilan Form Training

Pada *form Training* ini, akan dimasukkan data jenis stadium dari citra gambar X-Ray. Data yang dimasukkan akan disimpan ke dalam *database* sehingga dapat digunakan pada saat proses identifikasi jenis stadium penyakit kelenjar.

Untuk melakukan proses identifikasi jenis stadium penyakit kelenjar, maka *user* dapat mengklik *link* 'Pengujian', sehingga sistem akan menampilkan *form* Pengujian seperti terlihat pada gambar berikut:

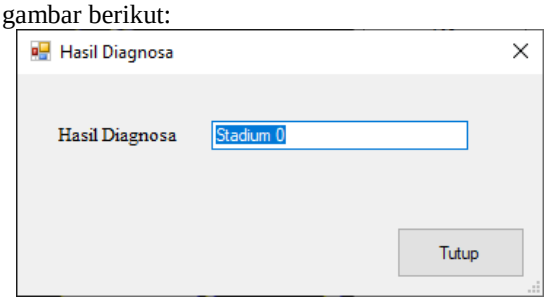


Gambar 13. Tampilan Form Pengujian

User harus memilih citra X-Ray yang akan diidentifikasi dan klik tombol Proses untuk memulai proses konversi ke citra *grayscale*. Setelah itu, klik tombol Lanjut sehingga sistem akan menampilkan *form* berikut:

Gambar 14. Tampilan Form Hasil Segmentasi

Form Hasil ini digunakan untuk menampilkan hasil pendeteksian jenis stadium penyakit kelenjar. Setelah itu, klik tombol Lanjut sehingga sistem akan menampilkan *form* Hasil seperti terlihat pada gambar berikut:



Gambar 15. Tampilan Form Hasil

5. KESIMPULAN

Setelah menyelesaikan pembuatan perangkat lunak ini, penulis dapat menarik beberapa kesimpulan sebagai berikut:

1. Jumlah sampel yang digunakan harus banyak. Hal ini diperlukan agar dapat meningkatkan akurasi hasil pendeteksian. Namun, jumlah sampel yang banyak akan mengakibatkan proses pendeteksian menjadi lama.
2. Metode *Region Growing* mampu mendeteksi jenis stadium kanker limfoma dengan

berdasarkan pada informasi dari *dataset* yang dimasukkan.

DAFTAR PUSTAKA

Felzenszwalb, P.F. dan D.P., Huttenlocher., ***Efficient Graph-Based Image Segmentation***, Institut Massachusetts dan Universitas Cornell, 2002.

Munir, R., **Pengolahan Citra Digital dengan Pendekatan Algoritmik**, Penerbit Informatika, Bandung, 2004

Purnomo, M. H. dan A. Muntasa, **Konsep Pengolahan Citra Digital dan Ekstrasi Fitur**, Graha Ilmu, Yogyakarta, 2010.

Ramadijanti, Nana dan Achmad Basuki, **Fitur Bentuk pada Citra**, 2008.

Wijaya, C. M. dan A. Prijono, **Pengolahan Citra Digital**, Informatika, Bandung, 2007.

Anonim, **Data dan Kondisi Penyakit Limfoma di Indonesia**, InfoDATIN (Pusat Data dan Informasi Kementerian Kesehatan RI), ISSN: 2442-7659, 2015.