IMPLEMENTASI ALGORITMA CONNECTED-LABELLING UNTUK MENDETEKSI OBJEK BINTANG PADA CITRA DIGITAL

Ericks Rachmat Swedia[1)], M. Ridwan Dwi Septian[2)], dan Margi Cahyanti[3)]
[1]Jurusan Sistem Komputer, Fakultas Ilmu Komputer dan Teknologi Informasi
[2]Jurusan Teknik Informatika, Fakultas Teknologi Industri
[3]Jurusan Sistem Informasi, Fakultas Ilmu Komputer dan Teknologi Informasi
Universitas Gunadarma
Jl. Margonda Raya 100, Depok, Jawa Barat
Telp : 021-78881112 (ext. 501)
Email : ericks_rs@staff.gunadarma.ac.id[1)], ridwandwiseptian@staff.gunadarma.ac.id[2)], margi@staff.gunadarma.ac.id[3)]

Abstrak

Bintang adalah objek yang menghasilkan cahaya sendiri, oleh karena itu bintang dapat terlihat jelas di malam hari. Proses mendeteksi bintang telah lama dilakukan oleh para astronom melalui teropong bintang yang dapat disimpan menjadi citra digital. Setiap bintang dalam sebuah citra digital terwakili oleh sekumpulan piksel yang saling terkoneksi dengan tingkat pencahayaan yang tinggi.
Penelitian ini membuat aplikasi pendeteksian bintang agar dapat mengetahui berapa jumlah objek bintang didalam citra berdasarkan tingkat cahaya dengan menggunakan HSL filter, menghitung banyaknya bintang menggunakan algoritma connected-labelling dari citra yang telah melalui proses grayscale, dan diimplementasikan dengan menggunakan bahasa pemrograman C#. Hasil uji coba menunjukkan aplikasi ini dapat digunakan untuk mendeteksi objek bintang pada citra. Kecepatan pemrosesan aplikasi dalam menghitung jumlah objek sangat ditentukan oleh besarnya ukuran matriks (lebar x tinggi) citra. Semakin kecil ukuran citra, semakin cepat aplikasi memproses perhitungan jumlah objek. Jumlah objek yang dapat terdeteksi di aplikasi ini, tergantung dari hasil HSL filter yang dilakukan, dalam penelitian ini HSL filter didapat dari nilai minimum histogram lightness, sehingga objek-objek yang memiliki nilai lightness dibawah nilai minimum histogram lightness dianggap noise dari citra dan tidak akan dideteksi

Kata kunci: *connected-labelling, HSL, deteksi, objek, bintang*

1. PENDAHULUAN

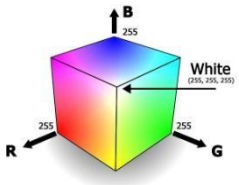
Bintang adalah objek yang menghasilkan cahaya sendiri, oleh karena itu bintang dapat terlihat jelas di malam hari. Proses mendeteksi bintang telah lama dilakukan oleh para astronom melalui teropong bintang. Astronom- astronom awal seperti Tycho Brahe berhasil mengenali 'bintang-bintang baru' di langit. Pada 1584 Giordano Bruno mengusulkan bahwa bintang-bintang sebenarnya adalah Matahari-matahari lain [1]. Pada abad berikutnya, ide bahwa bintang adalah Matahari yang jauh mencapai konsensus di antara para astronom. Menurut ilmu astronomi, definisi bintang adalah: "Semua benda masif (bermassa antara 0,08 hingga 200 massa matahari) yang sedang dan pernah melangsungkan pembangkitan energi melalui reaksi fusi nuklir", oleh sebab itu bintang katai putih dan bintang neutron yang sudah tidak memancarkan cahaya atau energi tetap disebut sebagai bintang. Bintang terdekat dengan Bumi adalah Matahari pada jarak sekitar 149,680,000 kilometer, diikuti oleh Proxima Centauri dalam rasi bintang *Centaurus* berjarak sekitar empat tahun cahaya [6].

Mendeteksi objek dalam sebuah citra telah lama menjadi bahasan di dunia pemrosesan citra. Penelitian ini membuat aplikasi untuk mendeteksi objek bintang yang terdapat dalam sebuah citra. Setiap bintang dalam citra terwakili oleh sekumpulan piksel yang saling terkoneksi dengan tingkat pencahayaan tinggi. Tingkat pencahayaan dalam sebuah citra hanya bisa dideteksi jika warna citra tersebut ditransformasikan ke bentuk ruang warna yang mempunyai nilai *lightness* atau *brightness* seperti ke model ruang warna HSL, dimana nilai *Lightness* dalam HSL mencerminkan skala pencahayaan dalam sebuah warna. Untuk mendapatkan objek dari sekumpulan *lightness* yang terkoneksi ini dapat digunakan algoritma *connected-labelling*, dari citra yang telah melalui proses *grayscale*.

2. PEMBAHASAN

2.1 RGB

Pada model warna RGB seperti yang dituliskan pada Gambar 1, setiap warna memperlihatkan komponen spectral primary red, green, dan blue. Model ini didasarkan pada sistem koordinat kartesian. *Sub-space* warna yang dicari adalah kubus dimana nilai RGB pada tiga sudut *cyan*, *magenta*, dan *yellow* ada pada tiga sudut lain, hitam adalah origin dan putih adalah titik paling jauh dari origin. Dalam model ini, grayscale (titik-titik nilai equal RGB) diperluas dari hitam ke putih, sepanjang garis gabungan dua titik. Perbedaan warna dalam model ini adalah titik-titik yang berada di dalam kubus dan didefinisikan oleh penyebaran vektor dari origin [2].



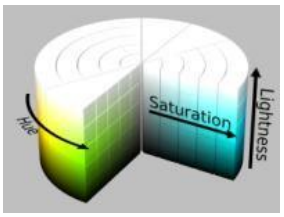
Gambar 1. Model Warna RGB

RGB sering digunakan pada aplikasi komputer, karena dengan ruang warna ini tidak diperlukan transformasi untuk menyampaikan informasi layar monitor. Alasan di atas juga menyebabkan RGB banyak dimanfaatkan sebagai ruang warna dasar bagi sebagian besar aplikasi [3].

2.2 HSL

Model HSL merupakan model yang ditemukan oleh Alvy Ray Smith pada tahun 1978. Model ini merepresentasikan warna dalam tiga komponen: *hue*, *saturation*, dan *lightness*. Secara konseptual HSL berbentuk kerucut berganda atau lingkaran dengan pucuknya berwarna putih, sudut dasarnya berwarna hitam, dan warna-warni sangat pekat pada sekeliling sisi lingkar horizontal serta pada bagian tengah warna abu-abu sedang. HSL model tersusun atas tiga karakteristik warna dasar [3]:

- *Hue* adalah warna yang direfleksikan atau pun ditransmisikan sebuah objek. Nilainya diukur dari lokasi pada roda standar warna, yang diekspresikan dengan nilai derajat sudut di antara 0° dan 360°. Dalam penggunaannya, *hue* mengidentifikasikan nama dari sebuah warna seperti merah, orange (jingga), atau hijau.
- *Saturation*, sering dikenal dengan chroma, yaitu ukuran atau kemurnian sebuah warna, *Saturation* merepresentasikan ukuran (kuantitas) dari proporsi keabuan pada *hue*, ukurannya dalam bentuk persentase dari 0% (*gray*) sampai dengan 100% (*fully saturated*). Pada roda standar warna, nilai *saturation* dari pusat roda (lingkaran) menuju tepian roda akan semakin bertambah.
- *Lightness* adalah sebuah ukuran relative skala pencahayaan (*lightness*) atau kegelapan (*darkness*) dari sebuah warna, umumnya diukur sebagai persentase dari 0% (*black*) sampai dengan 100% (*white*).



Gambar 2. Model Warna HSL

HSL dapat dianggap sebagai warna yang menggambarkan sebagai titik-titik dalam sebuah silinder (disebut warna solid) yang poros tengah berkisar dari hitam di bagian bawah untuk putih di bagian atas. Sudut sekitar sumbu terkait dengan "warna", jarak dari sumbu terkait dengan "kejenuhan", dan jarak sepanjang sumbu terkait dengan "ringan", "nilai" atau "terangnya"[3].

2.3 Grayscale

Citra *grayscale* adalah suatu citra yang terdiri dari warna hitam sebagai warna minimum dan warna putih sebagai warna maksimumnya, sehingga warna antara kedua warna (minimum dan maksimum),

tersebut adalah warna abu-abu. Untuk mengubah citra menjadi citra *grayscale* dapat dilakukan dengan menggunakan rumus berikut [4] :

$$I = 0.333R + 0.5G + 0.1666B \quad (1)$$

Dimana, R = Nilai piksel Red
G = Nilai piksel Green
B = Nilai piksel Blue

2.4 Transformasi RGB ke HSL

Untuk mentransformasi nilai dari RGB ke HSL diasumsikan koordinat-koordinat R, G, B [0,1] adalah berurutan merah, hijau, biru dalam ruang warna RGB. Nilai *max* adalah nilai maksimum dari nilai *red, green, blue*, dan min adalah nilai minimum dari nilai *red, green, blue*. Untuk memperoleh sudut *hue* [0,3600], nilai *saturation* [0,1] dan *lightness* [0,1] yang tepat untuk ruang warna HSL menggunakan rumus seperti berikut [3]:

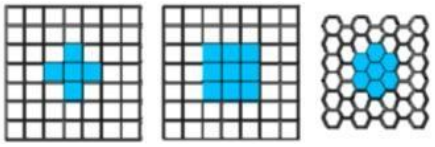
$$h\,(hue) = \begin{cases} 0, jika\ max = min \\ 60° \ x\ \left(\frac{G-B}{\max - min}\ mod\ 6\right), jika\ \max = R \\ 0° \ x\ \left(\frac{B-R}{\max - min} + 2\right), jika\ \max = G \\ 0° \ x\ \left(\frac{R-G}{\max - min} + 4\right), jika\ \max = B \end{cases} \quad (2)$$

$$l\,(lightness) = \frac{1}{2}\ (\max + \min) \quad (3)$$

$$s\,(saturation) = \begin{cases} 0\ ,\ jika\ \ \max = min \\ \frac{\max - min}{2L}\ ,\ \ jika\ \ l \leq 0.5 \\ \frac{\max - min}{2-(2L)}\ ,\ jika\ \ l > 0.5 \end{cases} \quad (4)$$

2.5 Algoritma Connected-Labelling

Algoritma *connected-labelling* digunakan sebagai pengenalan, *tracking*, identifikasi citra dan aplikasi kehidupan praktis lainnya. Sebuah piksel disebut bertetangga dengan piksel lain apabila piksel tersebut bertetangga langsung dengan piksel lain itu ataupun piksel lain itu merupakan tetangga dari tetangga piksel tersebut. Kriteria sebuah piksel merupakan tetangga dari piksel lain dapat berupa 4-*connectivity*, 6-*connectivity* ataupun 8- *connectivity* [5].



Gambar 3. 4, 8 dan 6-connectivity

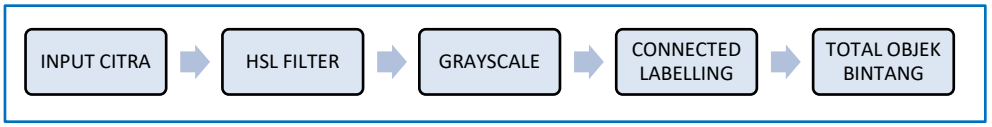
Piksel-piksel dalam region disebut *connected* (ada konektifitasnya atau keterhubungan) bila mematuhi aturan *adjacency* atau aturan "kedekatan" piksel . Aturan kedekatan piksel ini memanfaatkan sifat ketetanggaan piksel.

Dengan demikian piksel-piksel yang di katakan *connected* pada dasarnya memiliki sifat keterhubungan satu sama lain karena mereka masih memiliki hubungan *neighbourhood* atau ketetanggaan. Perlu diingat, bahwa citra yang bisa diolah dengan menggunakan metode ini adalah citra monokrom atau citra biner atau citra *grayscale*. Ketetanggaan harus memiliki panjang atau jarak 1 unit (langsung antara piksel dengan piksel tanpa ada perantara) [5].

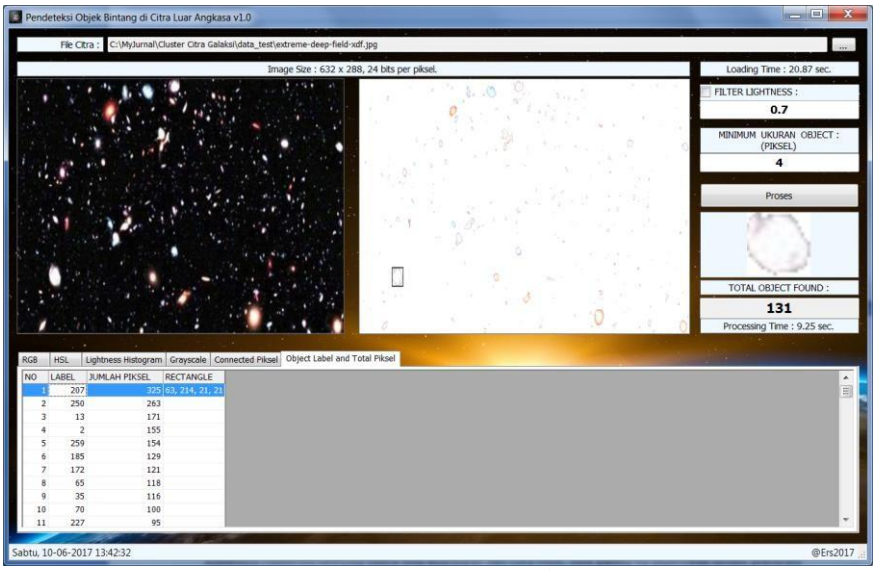
3. PERANCANGAN APLIKASI

Aplikasi didesain dengan pengguna memilih data citra yang akan dideteksi dan secara otomatis oleh aplikasi ditransformasi menjadi ruang warna HSL dan di *filter* nilai lightness-nya sesuai histogram

yang didapat, kemudian menampilkan citra hasil *filter* pada *interface* aplikasi. Pengguna selanjutnya dapat melakukan proses pendeteksian yang akan menjalankan algoritma *connected-labelling* dengan menekan tombol proses untuk menampilkan jumlah objek yang terdeteksi. Matriks citra setiap proses yang dihasilkan akan ditampilkan di aplikasi begitu juga dengan lama waktu pemrosesan. Rancangan *interface* dapat dilihat pada gambar 5, sedangkan alur aplikasi secara lengkap dapat dilihat pada gambar 4 dibawah ini,

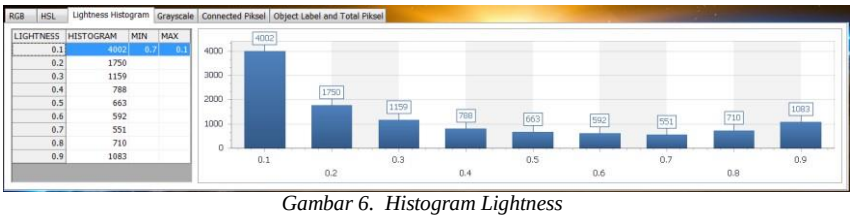


Gambar 4. Alur Aplikasi



Gambar 5. Rancangan Interface

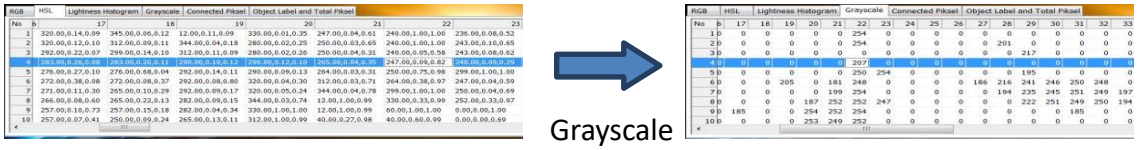
Langkah pertama setelah citra diinput adalah dengan mengubah citra menjadi HSL dan mem-*filter* nilai *lightness* berdasarkan nilai minimum histogram *lightness* yang didapat. Jika menginginkan nilai *filter lightness* yang lain, pengguna juga dapat memasukkan nilai *lightness* yang diinginkan.



Gambar 6. Histogram Lightness

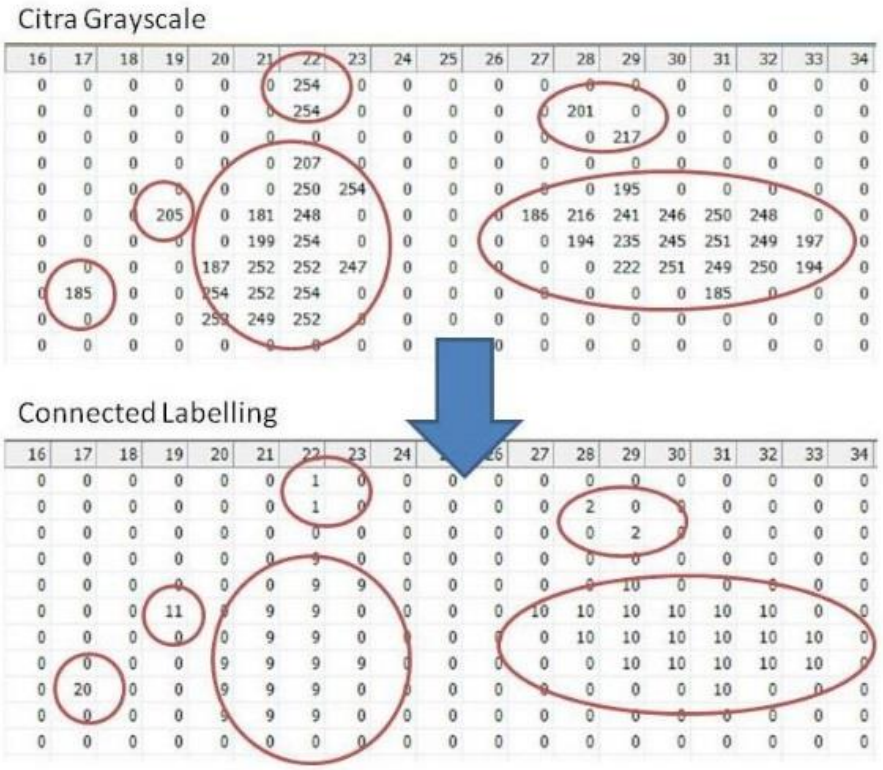
Gambar 7. Citra Hasil HSL filter dengan nilai lightness 0.7

Algoritma *connected-labelling* hanya bisa dijalankan dari citra yang hanya memiliki satu nilai dalam setiap piksel, oleh karena itu diperlukan proses *grayscale* untuk mengubah citra hasil HSL *filter* ke bentuk yang diinginkan.



Gambar 8. Hasil Grayscale

Algoritma *connected-labelling* berjalan dengan men-*scan* matriks citra dari kolom dan baris ke-0 hingga kolom dan baris terakhir. Proses ini akan memberi label saat menemukan nilai *gray* yang lebih besar dari 0, lalu mencari nilai *gray* sekelilingnya yang terhubung dan memberikan label yang sama. Proses ini kemudian berjalan ke kolom selanjutnya dan mengulangi langkah tersebut. Jika di kolom selanjutnya tidak ditemukan nilai *gray* yang lebih besar dari 0, proses ini akan menambah *counter label*.



Gambar 9. Hasil Algoritma Connected-Labelling

Berikut adalah hasil uji coba yang dilakukan dengan 3 buah citra.

No.	Citra Input (ukuran citra)	Output		
		HSL Filter	Objek	Lightness Filter
1.	 (480x270)		TOTAL OBJECT FOUND : **309** Processing Time : 6.61 sec.	0.7
2.	 (466x304)		TOTAL OBJECT FOUND : **28** Processing Time : 4.83 sec.	0.7
3.	 (632x288)		TOTAL OBJECT FOUND : **131** Processing Time : 8.68 sec.	0.7

Gambar 10. Hasil Uji Coba

4. SIMPULAN

Berdasarkan hasil uji coba yang dilakukan, dapat disimpulkan bahwa aplikasi ini dapat mendeteksi jumlah objek bintang yang terdapat dalam citra. Tingkat keakuratan aplikasi dalam mendeteksi objek berdasarkan HSL *filter* tergantung pada berapa nilai *lightness* yang ditetapkan. Skala *lightness* biasanya dimulai dari rentang 0.1 hingga 0.9. Jika nilai *lightness* ditetapkan terlalu rendah atau terlalu tinggi, akan ada objek yang tidak terdeteksi. Aplikasi menghitung nilai *lightness* ini berdasarkan nilai minimum dari histogram *lightness* citra. Dalam uji coba, aplikasi mendapatkan nilai *lightness* sebesar 0.7 yang merupakan nilai minimum dari histogram *lightness* citra, sehingga objek dengan *lightness* kurang dari 0.7 tidak akan dideteksi sebagai objek bintang.

Kecepatan aplikasi dalam menghitung jumlah objek sangat ditentukan oleh besarnya ukuran matriks citra (panjang x lebar), karena algoritma *connected-labelling* men-*scan* ukuran matriks (panjang x lebar) citra, semakin kecil ukuran panjang dan lebar citra, algoritma *connected-labelling* akan berjalan semakin cepat, secara umum algoritma *connected-labelling* memiliki waktu komputasi sebesar $O(n^2)$.

Aplikasi ini diharapkan berguna untuk mengenali objek bintang pada citra digital di sekumpulan database citra yang banyak (*big data*), sehingga proses pengenalan tersebut tidak lagi dihitung manual satu per-satu oleh manusia, namun dapat di-otomatisasi sehingga proses pengenalan objek bintang pada citra digital dapat berlangsung jauh lebih cepat.

5. DAFTAR RUJUKAN

[1] Drake, Stephen A. 2006. A Brief History of High-Energy (X-ray & Gamma-Ray) Astronomy.NASA HEASARC. [Online] (Updated 17 Agustus, 2006). Available at : https://heasarc.gsfc.nasa.gov/docs/heasarc/headates/heahistory.html. [Accessed 10 Mei 2017]

[2] Shapiro, Linda G. & Stockman, George C. 2002. *Computer Vision*. Prentice Hall.

[3] Swedia, Ericks Rachmat. & Cahyanti, Margi. 2015. *Algoritma Transformasi Ruang Warna*. Depok: Indie Publishing.

[4] Finlayson, G. D., Qiu, G., Qiu, M., 1999, *Contrast Maximizing and Brightness Preserving Color to Grayscale Image Conversion*. CGIV2008, 4th European Conference on Colour in Graphics, Imaging, and Vision.

[5] R. C. Gonzalez & R. E. Woods. 2008. *Digital Image Processing*. Upper Saddle River. NJ 07458.

[6] Hoskin, Michael.1998.*The Value of Archives in Writing the History of Astronomy*. Space Telescope Science Institute. [Online] (Updated 24 Agustus, 2006). Available at: http://www.stsci.edu/stsci/meetings/lisa3/hoskinm.html. [Accessed 10 Mei 2017]