ALGORITMA LABELING CITRA BINER DENGAN PERFORMANSI OPTIMAL PROCESSOR-TIME

Eril Mozef
Jurusan Teknik Elektro, Politeknik Negeri Bandung
e-mail: erilmozef@yahoo.com

ABSTRAK: Beberapa algoritma labeling untuk citra biner nxn yang diklaim optimal dalam literatur pada umumnya hanya optimal ditinjau dari aspek algoritmanya saja namun tidak optimal ditinjau dari dari aspek arsitektural. Disamping itu, kompleksitas-kompleksitas yang dihasilkan tersebut tidak murni karena masih mengandung konstanta yang tergantung harga n. Pada paper ini diperkenalkan suatu algoritma labeling dengan performansi optimal Processor-Time murni. Ini berarti optimal tidak hanya dicapai dari sisi algoritma namun juga dari sisi arsitektur dan murni karena kompleksitas yang didapat tidak mengandung konstanta yang tergantung harga n. Kompleksitas algoritma yang didapat tersebut adalah O(cn) dengan menggunakan O(n) prosesor. Pada paper ini diberikan pembuktian terhadap kompleksitas yang didapatkan dan perbandingan performansinya dengan beberapa algoritma yang ada.

Kata kunci : Pengolahan citra, *Labeling, Citra biner*, Performansi optimal *Processor-Time.*

ABSTRACT: *In the literature, some labeling algorithms of nxn binary image, which are generally claimed as optimal, are only optimal with regards to the algorithmic aspect but not to the architectural aspect. Aside from this, the complexities obtained are not pure because the constant still depends on the value of n. This paper presents a labeling algorithm, which reaches purely Processor-Time optimal performance. This means that the optimal performance is not only reached considering its algorithm but also its architecture and "pure" means that the complexity obtained does not depend on the value of n. The algorithm complexity obtained is O(cn) using O(n) number of processors. In this paper the justification of complexity obtained is given and the performance comparison with other existing algorithms is discussed.*

Keywords: *Image processing, Labeling, Binary image, Processor-Time optimal performance.*

1. PENDAHULUAN

Labeling adalah suatu proses pemberian label yang sama pada sekumpulan pixel pembentuk objek yang saling berdekatan pada suatu citra. Objek yang berbeda memiliki label yang berbeda pula. Labeling termasuk pemrosesan citra tahap *inter-mediate level*. Labeling memiliki peran yang sangat penting pada pengolahan citra untuk mempermudah proses penganalisaan bentuk dan pengenalan pola pada tahap *high level*.

Walaupun definisi labeling sudah cukup tua yang dimulai pada tahun 1966 [1], namun hingga kini pencarian terhadap algoritma labeling optimal terus berkembang dan masih menjadi topik yang menarik [8]. Penulis memperkirakan bahwa penelitian dibidang labeling akan terus berkembang beberapa puluh tahun kedepan sampai teknologi memungkinkan pengimplemen-tasian arsitektur paralel 2d dan 3d secara *massively* untuk pemrosesan citra dengan ukuran yang cukup signifikan.

Walaupun definisi labeling kelihatannya sederhana, namun labeling memiliki sifat yang *dependent* dan *regional*. Sifat *dependent* artinya pemberian label harus memperhitungkan pixel dan/atau label sebelumnya. Sedangkan sifat *regional* artinya pemberian label harus memper-hitungkan juga posisi pixel-pixel secara regional didalam citra.

Dengan sifatnya yang demikian maka secara alamiah labeling harus diproses secara sekuensial. Namun bila diproses secara sekuensial murni maka yang menjadi kendala adalah waktu pemrosesan yang sangat signifikan. Dengan sifatnya yang *dependent* dan *regional* itu pula maka

labeling tidak mungkin diproses secara paralel murni. Hal ini menjelaskan mengapa solusi labeling sebaiknya merupakan kombinasi dan kompromi antara solusi sekuensial dan paralel.

Karena masalahnya yang sederhana tapi solusinya yang tidak mudah itu dan karena sifatnya yang *dependent* dan *regional* serta melibatkan banyak aspek tersebut maka masalah labeling telah dijadikan sebagai salah satu tolok ukur pengujian performansi arsitektur paralel [14] [16].

Banyak algoritma labeling yang diusulkan dalam literatur. Literatur [8] mengupas *state-of-the-art* labeling ditinjau dari sisi algoritma dan arsitekturnya dan mulai dari solusi sekuensial sampai dengan paralel. Beberapa diantara algoritma-algoritma tersebut berhasil mencapai performansi optimal. Kompleksitas suatu algoritma dikatakan optimal bila selain dari kompleksitas tersebut tidak mungkin lagi didapatkan kompleksitas yang lebih kecil.

Sayangnya kebanyakkan kompleksitas-kompleksitas optimal yang dihasilkan tersebut hanya ditinjau dari sisi algoritmanya saja dan tidak memperhitungkan optimal ditinjau dari sisi arsitekturnya. Misalnya pada [10], diperoleh kompleksitas konstan labeling O(1) namun dengan menggunakan n^3 prosesor.

Pada paper ini diperkenalkan suatu algoritma labeling yang memiliki performansi optimal Processor-Time murni artinya optimal baik dari sisi algoritma maupun dari sisi arsitekturnya. Murni artinya konstanta kompleksitas algoritma tersebut tidak tergantung lagi harga n. Performansi ini sangat penting untuk dicapai mengingat biaya realisasi suatu arsitektur paralel adalah sangat tinggi. Dengan performansi optimal Processor-Time ini dimungkinkan tercapainya kondisi yang berimbang antara kecepatan dan harga.

2. BEBERAPA DEFINISI PENTING

Sebelum membahas algoritma, berikut ini diberikan beberapa definisi.

2.1 Performansi Optimal Processor-Time

Secara umum, performansi optimal Processor-Time untuk suatu permasalahan citra adalah suatu kondisi dimana perkalian antara jumlah processor yang digunakan dan kompleksitas algoritma yang didapat pada solusi paralel sama dengan perkalian antara jumlah processor yang digunakan dan kompleksitas algoritma optimal yang didapat pada solusi sekuensial (persamaan 1) [2] [10].

$$OPT_{(\text{masalah citra})} \longrightarrow P_p x T_p = P_s x T_s \qquad (1)$$

dimana:

Pp : jumlah prosesor pada struktur paralel.
Tp : kompleksitas algoritma paralel.
Ps : jumlah prosesor pada struktur sekuensial.
Ts : kompleksitas algoritma sekuensial.

Bila kita asumsikan bahwa solusi sekuensial menggunakan O (1) prosesor maka persamaan 1 dapat disederhanakan:

$$OPT_{(\text{masalah citra})} \longrightarrow P_p x T_p = T_s$$

Kompleksitas algoritma labeling sekuensial optimal telah berhasil mencapai $O(n^2)$ [8]. Dengan hasil ini maka persamaan 2 dapat ditulis:

$$OPT_{(\text{labeling})} \longrightarrow P_p x T_p = O\left(n^2\right) \qquad (2)$$

Catatan: Menurut Alnuweiri [8], kompleksitas $O(n^2)$ untuk labeling sekuensial ini didapat dengan menggunakan algoritma Union-Find dari Tarjan [14] (lihat bab algoritma sekuensial berbasis RAM).

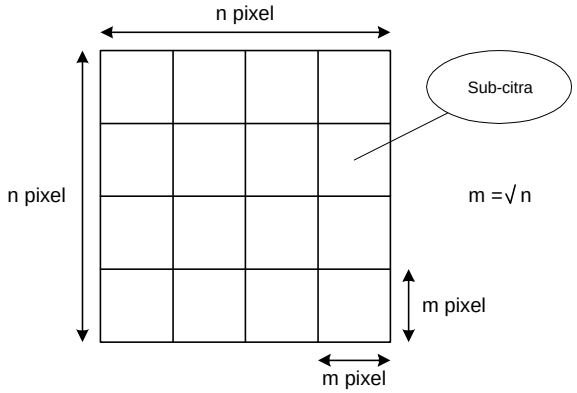
2.2 Scanning Dan Merging

Scanning adalah suatu proses penelusuran pixel-pixel untuk menganalisa suatu konfigurasi pixel dan label. *Scanning sisi* adalah proses penelusuran 2 buah sisi yang bersentuhan untuk mencari adanya 2 label yang berbeda.

Dalam hal terdapat 2 objek dengan label yang berbeda saling berdekatan dengan jarak 1 pixel maka dapat dilakukan proses *merging*. Mula-mula 2 pixel yang berdekatan dari kedua objek tersebut dibandingkan untuk mencari label yang terkecil (bisa juga yang terbesar bila

diinginkan) dari kedua label tersebut. Kemudian label yang terkecil ini dipergunakan untuk menggantikan nilai dari seluruh label yang terbesar.

2.3 Citra Dan Sub-citra



Gambar 1. Ukuran Citra dan Sub-Citranya

Citra biner adalah citra yang memiliki hanya 2 informasi yaitu:
Pixel 1 didefinisikan sebagai pixel objek.
Pixel 0 didefinisikan sebagai pixel background (non-objek).

Ukuran citra adalah nxn pixel yang terbagi dalam √n sub-citra (Gambar 1). Ukuran sub-citra adalah mxm, dimana m=√n.

Urutan indeks sub-citra sama dengan urutan indeks posisi pixel dan label yang dibahas berikut ini.

2.4 Indeks Posisi Pixel Dan Label

Pixel-pixel diberi indeks sesuai dengan posisi globalnya pada citra (bukan sub-citra). Urutan indeks disesuaikan secara urutan *raster-scan* yaitu dari kiri ke kanan dan dari atas ke bawah (Gambar 2).

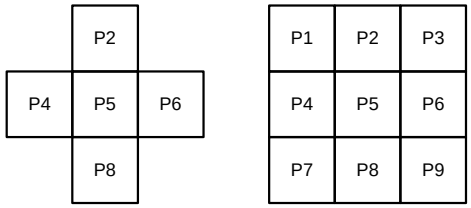
1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Gambar 2. Contoh indeks posisi pixel-pixel pada citra berukuran 4x4 pixel

Indeks posisi dimulai dari 1 dan diakhiri dengan 2^n atau $1 \leq$ indeks posisi $\leq 2^n$. Sedangkan indeks label dimulai dari 0 dan diakhiri dengan 2^n atau $0 \leq$ indeks label $\leq 2^n$. Indeks label antara $1 \leq$ indeks label $\leq 2^n$ digunakan untuk menandai pixel objek. Sedangkan indeks label 0 untuk pixel *background*.

2.5 Connexity

Operator lokal pixel untuk proses *scanning* citra yang telah dijelaskan dapat menggunakan operator lokal pixel 4-connexity atau 8-connexity (Gambar 3). Bila menggunakan prinsip 4-connexity maka 2 pixel yang bersinggungan secar diagonal dianggap 2 objek, sedangkan pada 8-connexity dianggap 1 objek (Gambar 4).



(a) Operator 4-connexity (b) Operator 8-connexity

Gambar 3. 2 jenis operator lokal pixel

0	0	1	0
1	0	1	0
1	0	0	1
0	1	1	1

(a) Citra biner

0	0	3	0
5	0	3	0
5	0	0	12
0	12	12	12

(b) Hasil labeling dengan 4-connexity

0	0	3	0
3	0	3	0
3	0	0	3
0	3	3	3

(c) Hasil labeling dengan 8-connexity

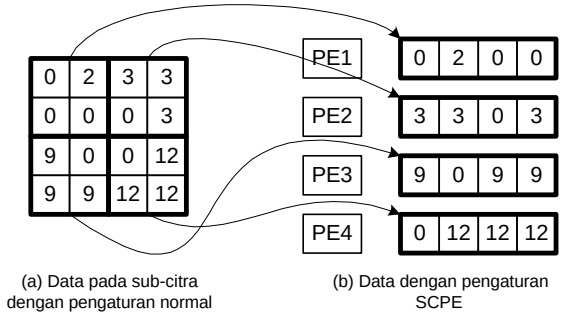
Gambar 4. Ilustrasi operasi labeling dengan 4 dan 8-connexity

Untuk mempermudah penjelasan, algoritma dianggap bekerja dengan prinsip 4-connexity. Namun tidak menutup kemungkinan algoritma bekerja pada 8-connexity. Adapun prinsip 4 dan 8-connexity dapat diilustrasikan pada Gambar 4.

2.6 Pengaturan Data Secara SCPE (Sub-Citra per PE)

Pengaturan data secara SCPE (Sub-citra per PE) adalah pengaturan data yang berada pada sebuah sub-citra dengan bentuk array 2D ke bentuk array 1D (Gambar 5). Indeks

data pada SCPE disesuaikan dengan urutan *raster-scan* dari data pada sub-citra [11].



Gambar 5. Data pada pengaturan normal dan pada pengaturan SCPE

3. BEBERAPA ALGORITMA LABELING YANG ADA DALAM LITERATUR

Dari sekian banyak algoritma labeling yang terdapat dalam literatur, berikut ini penulis berikan beberapa diantaranya yang berhubungan dengan algoritma labeling optimal Processor-Time yang diusulkan.

3.1 Algoritma Labeling Sekuensial (Berbasis RAM) Dengan Kompleksitas $O(c_{(n)}\, n^2)$

Algoritma labeling sekuensial berbasis RAM (Random Access Memory) ini adalah algoritma yang paling tua keberadaannya yaitu bersamaan dengan lahirnya definisi labeling [1].

Algoritma terdiri dari 2 tahap:

Tahap 1: Mula-mula pixel demi pixel citra di-*scan* secara *raster* oleh suatu operator pixel 3x3 pixel. Operator ini dapat mendeteksi keadaan pixel yang sedang diexaminasi (*current* pixel) antara lain:

1) pixel awal: kondisi dimana current pixel tidak memiliki pixel tetangga yang terlewati pada proses *scanning* sebelumnya.
2) pixel sambung: kondisi dimana *current* pixel memiliki sebuah pixel tetangganya yang terlewati pada proses *scanning* sebelumnya dan yang telah memiliki label.
3) pixel gabung: kondisi dimana *current* pixel memiliki 2 pixel tetangga yang keduanya telah terlewati pada proses *scanning* sebelumnya dan yang telah memiliki label.

Bila pixel awal ditemukan maka label baru diberikan kepada *current* pixel. Bila tidak maka bila merupakan pixel sambung maka label dari pixel tetangganya tersebut diberikan kepada *current* pixel. Bila tidak, bila pixel gabung ditemukan maka lakukan proses *merging* LUT (Look-Up-Table) (ke tahap 2). Bila tidak maka pindahkan operator pixel ke pixel berikutnya.

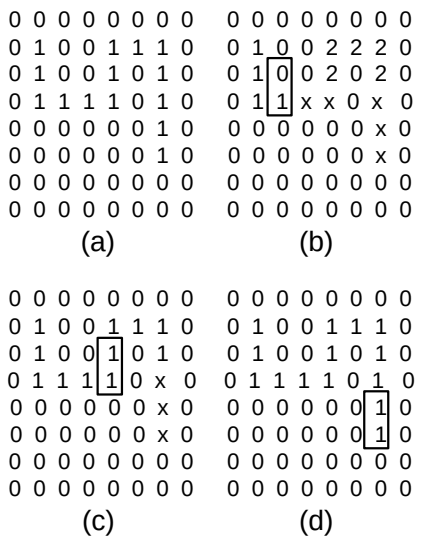
Tahap2: Proses *merging* LUT ini prinsipnya adalah mencari, pada seluruh alamat yang ada, label yang sama dengan label yang terkecil pada proses perbandingan 2 label (lihat definisi *merging*). Bila kita asumsikan pada LUT terdapat $O(n^2)$ pixel gabung (dalam hal objek kompleks) sedangkan *merging* sebuah pixel gabung memerlukan waktu $O(n^2)$ pada memori bertipe RAM. Jadi masalah ini bila dikerjakan dengan algoritma sekuensial sederhana memerlukan waktu $O(n^4)$.

Kebanyakan algoritma yang ada dalam literatur memberikan alternatif *merging* yang lebih cepat. Salah satu solusi yang terbaik adalah algoritma Tarjan [14] yang memungkinkan mereduksi kompleksitas *merging* LUT menjadi $O(c_{(n)}n^2)$, dimana c(n) adalah konstanta yang merupakan kebalikan dari fungsi Ackerman yang naik secara perlahan. Perlu dicatat bahwa konstanta ini masih mengandung harga n.

3.2 Algoritma Labeling Sekuensial (Berbasis CAM) Dengan Kompleksitas $O(n^2)$

Algoritma labeling ini secara umum hampir sama dengan yang berbasis memori RAM bedanya terdapat pada tahap 2 dimana proses *merging* label dilakukan pada memori CAM. Sesuai dengan namanya, memori CAM dapat dialamati secara isinya yang memungkinkan seluruh isi yang memiliki data yang sama diganti dengan data yang lain hanya dalam waktu konstan O(1). Untuk lebih detilnya proses penulisan CAM ini dapat dilihat pada [3] [16] [17] [18]. Bila isi dari CAM ini dianggap label,

maka untuk mengganti label-label yang sama yang terdapat di n^2 lokasi hanya memerlukan waktu konstan O(1).



Gambar 6. Algoritma labeling sekuensial $O(n^2)$ dengan memori CAM

Jadi bila terdapat n^2 buah pixel gabung dimana merging 1 pixel gabung memerlukan waktu O(1) maka kompleksitas *merging* dengan CAM adalah O(n). Sehingga total kompleksitas ditentukan hanya dari waktu *scanning* pixel yaitu $O(n^2)$ (Gambar 6). Catatan bahwa konstanta yang terkandung pada kompleksitas ini adalah murni konstan dan tidak mengandung lagi harga n seperti halnya pada labeling yang berbasis algoritma Tarjan.

Labeling sekuensial dengan pendekatan CAM (Content Addressable Memory) untuk penyelesaian *merging* diperkenalkan dalam [16].

3.3 Algoritma Labeling Paralel 1D Processor-Time Optimal Dengan Kompleksitas O(c(n) n)

Dengan menggunakan teknik pengaturan data secara SCPE yang telah dibahas sebelumnya (Gambar 5) adalah mungkin mengoptimalkan kompleksitas yang didapat dengan arsitektur paralel 1D [10].

Mula-mula citra dengan ukuran nxn pixel dibagi menjadi n buah sub-citra masing-masing berukuran $\sqrt{n}$ x $\sqrt{n}$ pixel (Gambar 1). Masing-masing sub-citra ini kemudian diolah oleh sebuah PE (Gambar 5).

Algoritma terdiri dari 3 tahap:

Tahap 1: Setiap sub-citra diberi label dengan menggunakan teknik sekuensial konvensional dan algoritma Union-Find Tarjan [14] yang memungkinkan didapatkan kompleksitas $O(k^2c(k))$ dimana c(k) adalah kebalikan dari fungsi Ackerman yang naik secara perlahan. Jika diambil $k=\sqrt{n}$ maka c(k) dianggap konstan. Sehingga tahap ini dianggap mendapatkan kompleksitas O(n).

Tahap 2: Tahap ini adalah tahap *merging*. *Merging* dilakukan terhadap label-label yang berada pada sisi-sisi dari 4 sub-citra yang bertetanggaan sehingga untuk 2 buah pixel yang berdekatan dengan label yang berbeda akan menjadi sama dengan memilih label terkecil dari ke 2 label tersebut. Kemudian proses yang sama dilakukan untuk 4 sub-citra yang lebih besar dan begitu seterusnya sampai seluruh sisi-sisi sub-citra ter-*merging*. Secara total ada log ($\sqrt{n}$) *merge*. Karena setiap merge membutuhkan $2^{2f} + 2^f\sqrt{n}$ iterasi dimana $1 \le f \le \log(\sqrt{n})$, maka kompleksitas tahap ini adalah O(n).

Tahap 3: Setiap label pada sisi-sisi yang telah di-*merging* kemudian dipropagasikan kedalam sub-citra. Kompleksitas tahap ini adalah O(n).

Jadi kompleksitas total algoritma ini adalah O(n) namun dengan konstanta kompleksitas yang masih tergantung dari n.

3.4 Algoritma Labeling Paralel 2D Dengan Kompleksitas O(n)

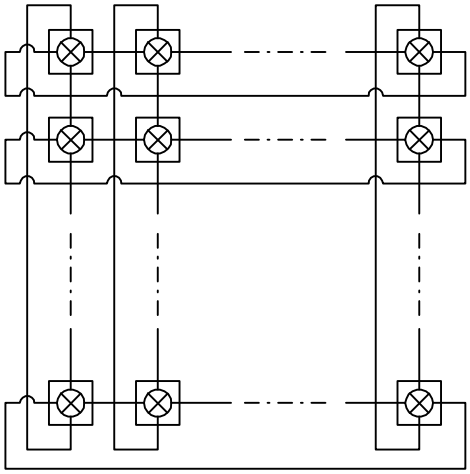
Algoritma ini menggunakan teknik *divide-and-conquer* yang diimplementasikan pada arsitektur Polymorphic-Torus (Gambar 7), arsitektur 2D dengan n^2 PE dengan bus yang dapat direkonfigurasi [15]. Perlu dicatat bahwa arsitektur paralel 2D pada umumnya memiliki sifat asosiatif antara prosesornya sehingga memiliki kemampuan *merging* dalam waktu konstan O(1) seperti halnya CAM.

Algoritma ini terdiri dari 2 tahap:

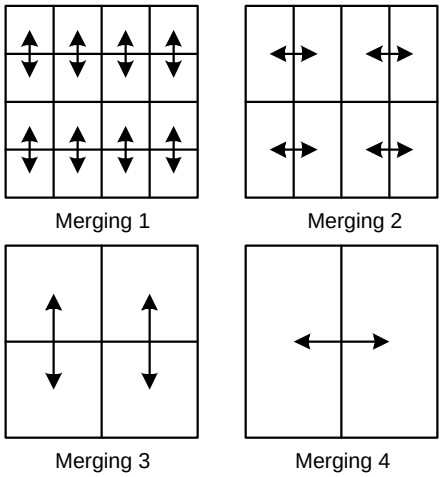
Tahap 1: Setiap pixel objek (bukan *background*) diberi sebuah label sesuai dengan

indeks posisinya pada citra (Gambar 2). Ini dilakukan dengan kompleksitas O(1). Hal ini dimungkinkan karena setiap pixel berhubungan dengan satu PE.

Tahap 2: Mula-mula area-area dengan ukuran terkecil 1x1 pixel di-*merging* secara paralel untuk mendapatkan area 2x1. Kemudian dilanjutkan dengan me*merging* area-area berukuran 2x1 untuk mendapatkan area yang lebih luas 2x2. Proses *merging* ini diulang untuk mendapatkan area-area berukuran 2x2, 4x2, 4x4 dan seterusnya sampai mendapatkan area terbesar dengan nxn pixel (Gambar 8). Kompleksitas dari tahap ini hanya ditentukan dari banyaknya pixel pada sisi area yang terpanjang yang dalam hal ini n sehingga terdapat n kali *merging*. Bila proses merging dilakukan dalam waktu O(1) maka kompleksitas total dari algoritma ini adalah O(n) dengan konstanta murni yang tidak tergantung dari n.



Gambar 7. Arsitektur paralel 2D Polymorphic-Torus

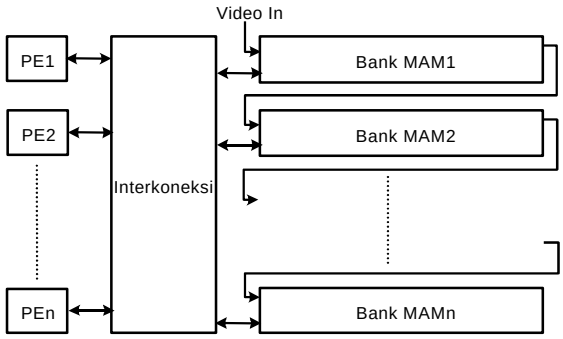


Gambar 8. Algoritma paralel O(n) pada arsitektur 2D Polymorphic-Torus

4. ALGORITMA PARALEL 1D PROCESSOR-TIME OPTIMAL DENGAN KOMPLEKSITAS MURNI O(n) YANG DIUSULKAN

Untuk dapat menjelaskan algoritma labeling yang dikembangkan, penulis terlebih dahulu menggambarkan model arsitektur paralel yang digunakan.

4.1 Model Arsitektur Paralel yang Digunakan



Gambar 9. Model arsitektur paralel yang digunakan

Arsitektur yang digunakan adalah berjenis 1d dengan n PE (Processor Element), n bank memori MAM dan sebuah jaringan interkoneksi (Gambar 9). Model ini adalah generalisasi dari arsitektur spesifik untuk proses labeling citra yang telah kami kembangkan sebelumnya [4] [5] [7] [9].

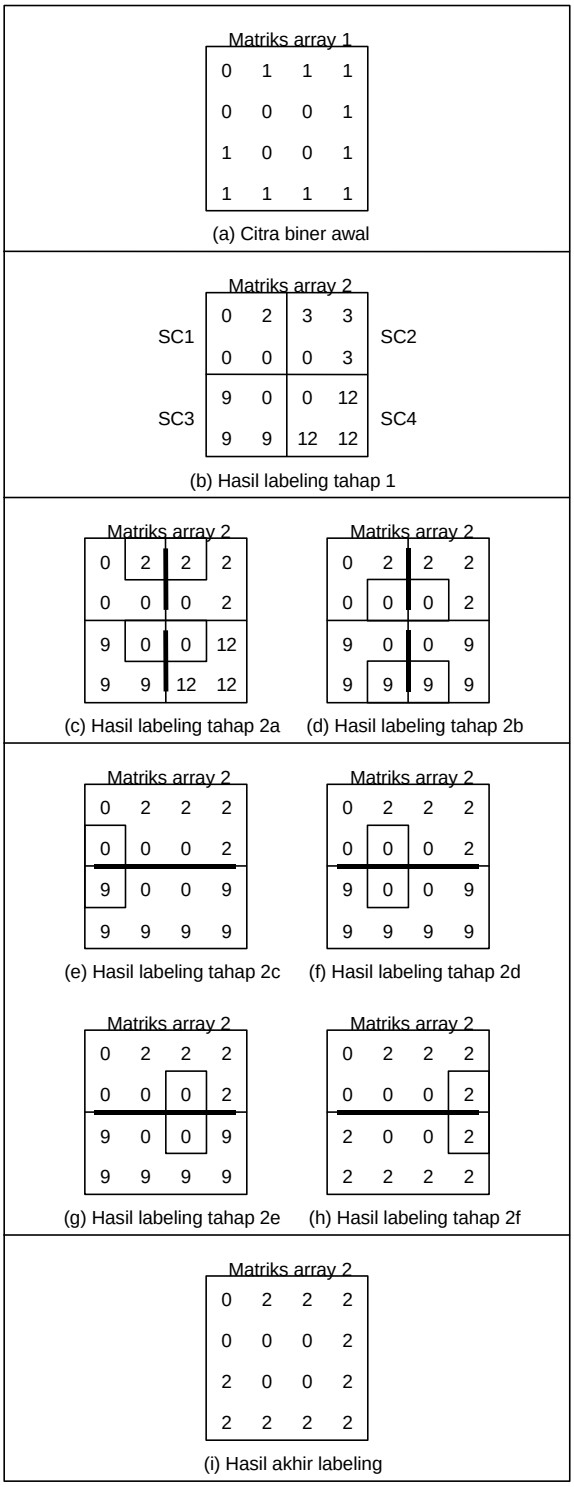
Secara umum model ini dapat dijelaskan sebagai berikut. Sebuah bank memori MAM apapun melalui jaringan interkoneksi dapat diakses oleh sembarang PE. Begitupun sejumlah k bank MAM ($0<k\leq n$) dapat diakses oleh sejumlah k PE ($0<k\leq n$). Pada mode tulis, satu atau beberapa bank dapat ditulis oleh sebuah PE. Pada mode baca, satu atau beberapa PE dapat membaca sebuah bank MAM.

Memori MAM memiliki fungsi RAM dan CAM yang memungkinkan penggantian berapapun jumlah label yang sama dengan label lain hanya dalam waktu O(1).

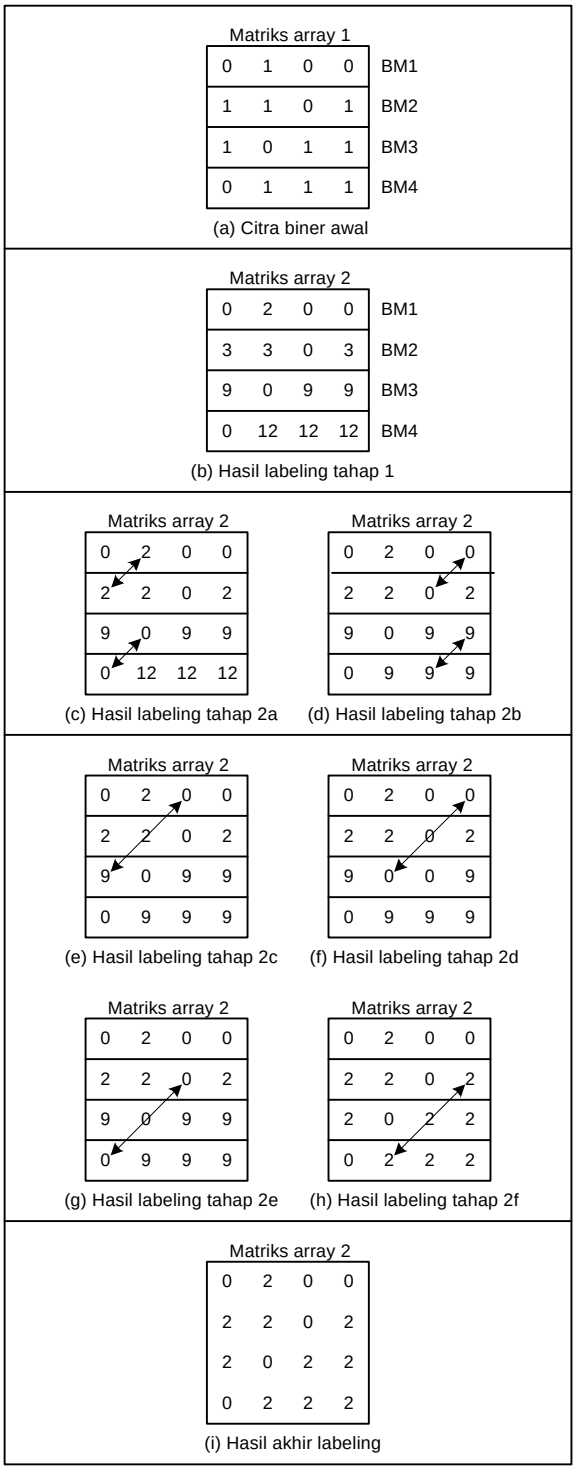
4.2 Algoritma Yang Dikembangkan

Pada Gambar 10 dan Gambar 11 dijelaskan algoritma labeling optimal Processor-

Time yang diusulkan untuk n=4. Algoritma dapat dijelaskan baik secara 2D dengan pengaturan data secara normal (Gambar 10) maupun 1D dengan pengaturan data SCPE (Gambar 11). Penjelasan cara pertama merupakan sekedar ilustrasi untuk mempermudah pengertian algoritma. Sedangkan penjelasan cara kedua adalah sesuai dengan keadaan sebenarnya.



Gambar 10. Ilustrasi labeling optimal Processor-Time O(n) yang diusulkan pada struktur array berdimensi 2



Gambar 11. Ilustrasi labeling optimal Processor-Time O(n) yang diusulkan pada struktur array berdimensi 1

Algoritma terdiri dari 2 tahap:

Tahap 1: Pada tahap ini dilakukan proses labeling pada setiap sub-citra secara sekuensial. Tahap ini dapat mengacu pada

algoritma sekuensial berbasis CAM yang telah dibahas sebelumnya karena memiliki kesamaan. Perbedaannya adalah bahwa disini yang diproses adalah sub-citra dengan ukuran √n x √n pixel (Gambar 5). Hasil dari tahap ini dapat dilihat pada Gambar 10b atau Gambar 11b.

Kompleksitas dari tahap ini merupakan jumlah maksimum pixel yang di-*scan* pada sub-citra yaitu √n x √n. Jadi kompleksitas tahap ini adalah O(n).

Tahap 2: Pada tahap ini dilakukan proses *scanning* hanya pada 2 sisi sub-citra yang bersentuhan. Kemudian dilakukan proses *merging*. Mula-mula scanning dan merging ini dilakukan pada setiap 2 sub-citra yang terkecil secara paralel. Misalnya pada contoh yaitu SC1-SC2 dan SC3-SC4 Gambar 10c dan d atau Gambar 11c dan d. Kemudian dilanjutkan dengan *merging* 2 area dari hasil *merging* sebelumnya. Proses *merging* diulang sampai didapat area berukuran nxn pixel (Gambar 10h atau Gambar 11h). Hasil akhir labeling ditunjukkan pada Gambar 10i atau Gambar 11i.

Kompleksitas dari tahap ini dapat dihitung dari banyaknya jumlah *scan* yang terjadi. Jumlah *scan* ini sama dengan banyaknya jumlah pixel pada seluruh sisi yang di-*merging* (NbPixelSisi).

$$\text{NbPixelSisi} = \sqrt{n} + [2\sqrt{n} + 2\sqrt{n} + 4\sqrt{n} + 4\sqrt{n} + \ldots + (\sqrt{n}\sqrt{n})/4 + (\sqrt{n}\sqrt{n})/4 + (\sqrt{n}\sqrt{n})/2 + (\sqrt{n}\sqrt{n})/2] + n \quad (3)$$

Persamaan 4 dapat ditulis dengan:

$$\text{NbPixelSisi} = \sqrt{n} + R + n \quad (4)$$

dimana:

$$R = 2\sqrt{n} + 2\sqrt{n} + 4\sqrt{n} + 4\sqrt{n} + \ldots + (\sqrt{n}\sqrt{n})/4 + (\sqrt{n}\sqrt{n})/4 + (\sqrt{n}\sqrt{n})/2 + (\sqrt{n}\sqrt{n})/2 \quad (5)$$

Persamaan 6 dapat disederhanakan:

$$R = 4\sqrt{n}(2^0 + 2^1 + + 2^i + \ldots + 2^{I-i} + 2^I) \quad (6)$$

Karena $2^I = 4\sqrt{n}$ atau $I = \log(4\sqrt{n})$, persamaan diatas dapat ditulis menjadi:

$$R = 4\sqrt{n \sum_{i=0}^{\log(4\sqrt{n})} \left(2^i\right)} \quad (7)$$

karena: $\sum_{i=0}^{\log(4\sqrt{n})} \left(2^i\right) = 2^{\left[\log(4\sqrt{n})\right]+1} - 1$ (8)

Jadi:

$$R = \sqrt{n\left(2^{\left[\log(4\sqrt{n})+1\right]} - 1\right)} \quad (9)$$

Dengan mengetahui $2^{\left[\log(4\sqrt{n})\right]+1} = 2\sqrt{\frac{n}{4}}$ maka:

$$R = 2n - 4\sqrt{n} \quad (10)$$

Dengan menggabungkan persamaan 5 dan 10 maka didapat:

$$\text{NbPixelSisi} = \sqrt{n + \left(2n - 4\sqrt{n}\right) + n}$$

jadi:

$$\text{NbPixelSisi} = 3\left(n - \sqrt{n}\right) \quad (11)$$

Jadi banyaknya pixel sisi adalah 3(n-√n) yang berarti pula terdapat 3(n-√n) *merging*. Bila sebuah *merging* memerlukan waktu O(1) maka total waktu yang diperlukan untuk merging adalah O(3(n-√n)). Karena n jauh lebih besar dari pada √n jadi kompleksitas dari tahap 2 ini adalah O(n) dengan konstanta murni sama dengan 3.

Dari hasil perhitungan tahap 1 dan 2 maka kompleksitas algoritma labeling yang diusulkan adalah O(n) dengan konstanta murni adalah 3.

5. HASIL DAN DISKUSI

Pada Tabel 1 diberikan perbandingan performansi dari beberapa algoritma yang telah dibahas. P adalah jumlah prosesor, T adalah waktu yang diwakili dengan kompleksitas algoritma. K adalah konstanta kompleksitas. PxT adalah hasil perkalian antara P dan T. PTs=PTp adalah kondisi dimana PxT paralel = PxT sekuensial. Perf PT adalah performansi Prosesor-Time yang dicapai. Algoritma Sekuensial menjadi acuan untuk menentukan performansi optimal Processor-Time, jadi pada kolom PTp = PTs dan Perf. PT, bagian ini tidak diisi (diberi tanda X pada Tabel 1).

Tabel 1. Perbandingan performansi optimal dan performansi optimal Processor-Time dari beberapa algoritma labeling

Algo	P	T	K	PxT	PTp=PTs	Perf. PT

Sek RAM	1	$O(n^2)$	c(n)	$O(n^2)$	X	X
Sek CAM	1	$O(n^2)$	c	$O(n^2)$	X	X
Par 1D RAM	n	O(n)	c(n)	$O(n^2)$	Ya	Optimal tdk murni
Par 2D	n^2	O(n)	c	$O(n^3)$	Tidak	Tidal optimal
Par 1D CAM	n	O(n)	c	$O(n^2)$	Ya	Op murni

Algoritma labeling berbasis RAM baik itu sekuensial maupun paralel menghasilkan kompleksitas dengan konstanta c yang masih tergantung dari n. Hal ini menyebabkan algoritma paralel berbasis RAM memiliki performansi optimal Processor-Time yang tidak murni. Sedangkan algoritma labeling berbasis CAM baik itu sekuensial maupun paralel menghasilkan kompleksitas dengan konstanta c yang tidak lagi tergantung dari n. Hal ini menyebabkan algoritma paralel berbasis CAM yang penulis usulkan memiliki performansi optimal Processor-Time yang murni (bagian yang diarsir pada Tabel 1).

Performansi optimal Processor-Time yang penulis berhasil kembangkan ini, sepengetahuan penulis belum pernah ada sebelumnya dalam literatur. Dengan jumlah prosesor O(n) penulis berhasil mendapatkan kompleksitas murni O(n). Sebelumnya Alnuweiri [11] mengklaim telah berhasil mencapainya namun hasil ini tidak murni dikarenakan kompleksitas algoritma yang didapat O(n) masih mengandung konstanta perkalian yang tergantung harga n walaupun sangat kecil. Jadi sebenarnya kompleksitas tersebut lebih tepatnya dapat ditulis dengan $O(c_{(n)}n)$.

6. KESIMPULAN

Pada paper ini telah dibahas suatu algoritma labeling objek pada citra biner berukuran nxn pixel dengan performansi optimal Processor-Time dimana baik kompleksitas algoritma yang dihasilkan maupun jumlah prosesor yang digunakan adalah optimal. Dengan jumlah prosesor O(n) didapatkan kompleksitas algoritma O(cn) dengan konstanta c murni (yang tidak bergantung lagi pada harga n).

Dengan hasil ini, maka algoritma kami merupakan salah satu algoritma labeling pertama dengan performansi optimal Processor-Time murni karena selama ini performansi Processor-Time yang terdapat dalam literatur masih mengandung konstanta kompleksitas c yang masih tergantung pada harga n. $O(c_{(n)}n)$

Performansi optimal Processor-Time sangat penting untuk dicapai mengingat biaya realisasi suatu arsitektur paralel adalah sangat tinggi. Dengan performansi optimal Processor-Time ini adalah mungkin untuk mencapai kondisi yang berimbang antara kecepatan dan harga.

DAFTAR PUSTAKA

1. A. Rosenfeld, et al. *Sequential Operations in Digital Picture Processing.* Journal of the Association for Computing Machinery, vol. 13, no. 4, 1966, pp. 471-494.

2. E. Mozef. *Arsitektur Paralel Pengolahan Citra dan Performansi Optimal.* Prosiding Ilmu Komputer dan Teknologi Informasi III (SNKK3), Agustus 2002, Jakarta, pp. 39-44.

3. E. Mozef. *Memory MAM (Multi-mode Access Memory) untuk Pengolahan Citra Paralel: Prinsip, Aplikasi dan Performansi.* Jurnal Teknik Elektro, Universitas Kristen Petra, Oktober 2002.

4. E. Mozef, et al. *Real-time connected component labeling on one-dimensional array processors based on Content-Addressable Memory: optimization and implementation.* UMIST-IEEE 3rd International Workshop on Image and Signal Processing, Manchester, United Kingdom, Nov. 96, pp. 691-694.

5. E. Mozef, et al. *Parallel architecture dedicated to connected component analysis.* IAPR-IEEE 13th International Conference on Pattern Recognition, Vienna, Austria, August 96, pp. 699-703. (IEEE Computer Society Press).

6. E. Mozef, et al. *Parallel architecture dedicated to connected component labelling in O(n log n): FPGA Implementation.* SPIE International Symposium on Las., Opt., and Vision for

Product. In Manufact. II, Micropolis, Besancon, France, June 96, pp. 120-125.

7. E. Mozef, et al. *Architecture dediee a l'algorithme parallel O(n log n) d'etiquetage de composantes connexes.* 3eme Journee Adequation Algorithme Architecture en Traitement du Signal et Images, Toulouse, France, Jan. 96, pp. 83-89. (In collaboration with IEEE signal processing).

8. H.M. Alnuweiri, et al. *Parallel Architectures and Algorithms for Image Component Labeling.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14, no. 10, Oct., 1992, pp. 1014-1034.

9. H. M. Alnuweiri, et al. *Processor-Time Optimal Parallel Algorithms for Digitized Images on Mesh-Connected Processor Arrays.* Algorithmica, vol. 6, 1991, pp.698-733.

10. H.M. Alnuweiri. *Constant Time Parallel Algorithms for Image Labeling on a Reconfigurable Network of Processors.* IEEE Trans. on Parallel and Distributed Systems, vol. 5, no. 3, Mar., 1994, pp. 320-326.

11. H. M. Alnuweiri, et al. *Optimal Image Computations on Reduced Processing Parallel Architectures.* Parallel Architectures and Algorithms for Image Understanding, New York: Academic Press, 1991, pp. 157-183.

12. R.E. Tarjan. *Efficiency of a Good But Not Linear Set Union Algorithm.* Journal of the Association for Computing Machinery, vol. 22, Apr., 1975, pp. 215-225.

13. C. Weems, et al. *The DARPA Image Understanding Benchmark for Parallel Computer.* J. Parallel Distributed Computing, vol. 11, no. 1, Jan., 1991, pp.1-24.

14. K. Preston. *The Abindon Cross Benchmark Survey.* IEEE Computer, July, 1989, pp. 9-18.

15. H. Li, et al. *Polymorphic-Torus architecture for computer vision.* IEEE Trans. on Patt. Analysis and Machine Intelligence, vol. 11, no.3, Mar., 1989, pp. 233-243.

16. W.E. Snyder, et al. *Content-Addressable read/write memories for image analysis.* IEEE Transactions on Computers, vol. C-31, no. 10, Oct., 1982, pp. 963-968.

17. Y.C. Shin, et al. *A special purpose Content-Addressable Memories chip for real-time image processing.* IEEE Journal of Solid-State Circuits, vol. 27, no. 5, Mai, 1992, pp. 737-744.

18. Y. Fujino, et al. *Facial image tracking system architecture utilizing real-time labeling.* Proc. SPIE-Int. Soc. Opt. Eng., vol. 2094, no. Pt.1, 1993, pp.2-11.