TUGAS AKHIR – KI141502

IDENTIFIKASI SEL DARAH MERAH BERTUMPUK MENGGUNAKAN UNSUPERVISED BAYESIAN CLASSIFICATION PADA CITRA MIKROSKOPIK SEL DARAH

RM REZA RIZKY PRATAMA
NRP 5110 100 181

Dosen Pembimbing I
Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.

Dosen Pembimbing II
Diana Purwitasari, S.Kom., M.Sc.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2015

UNDERGRADUATE THESIES – KI141502

OVERLAPPED RED BLOOD CELL IDENTIFICATION OF BLOOD MICROSCOPIC IMAGE USING UNSUPERVISED BAYESIAN CLASSIFICATION

RM REZA RIZKY PRATAMA
NRP 5110 100 181

First Advisor
Dr. Eng.Chastine Fatichah, S.Kom., M.Kom.

Second Advisor
Diana Purwitasari, S.Kom., M.Sc.

DEPARTMENT OF INFORMATICS
Faculty of Information Technology
Sepuluh Nopember Institute of Technology
Surabaya 2015

LEMBAR PENGESAHAN

IDENTIFIKASI SEL DARAH MERAH BERTUMPUK MENGGUNAKAN UNSUPERVISED BAYESIAN CLASSIFICATION PADA CITRA MIKROSKOPIK SEL DARAH

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Cerdas Visual
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh
RM REZA RIZKY PRATAMA
NRP : 5110 100 181

Disetujui oleh Dosen Pembimbing Tugas Akhir:

1. Dr. Eng. Chastine Fatichah, S.Kom., M.Kom. .....
   (NIP: 19751220 200112 2 002) (Pembimbing 1)

2. Diana Purwitasari, S.Kom., M.Sc .....
   (NIP: 19780410 200312 2 001) (Pembimbing 2)

SURABAYA
JULI, 2015

IDENTIFIKASI SEL DARAH MERAH BERTUMPUK MENGGUNAKAN UNSUPERVISED BAYESIAN CLASSIFICATION PADA CITRA MIKROSKOPIK SEL DARAH

Nama Mahasiswa	**:**	**RM Reza Rizky Pratama**
NRP	**:**	**5110 100 181**
Jurusan	**:**	**Teknik Informatika FTIF-ITS**
Dosen Pembimbing 1	**:**	**Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.**
Dosen Pembimbing 2	**:**	**Diana Purwitasari, S.Kom., M.Sc.**

Abstrak

Jumlah sel dapat digunakan untuk menentukan jenis penyakit seperti anemia disebabkan kurangnya sel darah merah atau leukimia disebabkan lebihnya sel darah putih, dan lain-lain. Dalam proses ekstrasi sel secara otomatis dari data citra mikroskopik sel darah, salah satu masalah adalah terdapatnya sel bertumpuk yang mengakibatkan ketidakakuratan penghitungan jumlah sel. Pada Tugas Akhir ini diimplementasikan identifikasi sel darah merah bertumpuk dalam citra mikroskopik sel darah dengan unsupervised Bayesian classification.

Pemisahan sel yang bertumpuk akan dimodelkan sebagai suatu cluster. Kemudian digunakan algoritma Expectation-Maximization untuk mendapatkan hasil parameter yang optimal sebagai nilai masukan Bayesian classifier. Cluster validity index digunakan untuk estimasi jumlah sel bertumpuk.

Pada uji coba penghitungan sel darah merah bertumpuk menggunakan metode unsupervised bayesian classification dilakukan perbandingan dengan penghitungan secara manual. Indikator undersegmented dan oversegmented menunjukkan bentuk dari sel darah merah yang tidak tepat bertumpuk. Uji coba metode mampu memberikan hasil cukup memuaskan dengan rata-rata akurasi 87,94%, serta rata-rata

error undersegmented dan oversegmented sebesar 3,82% dan 8,24%.

Kata Kunci: Sel Bertumpuk, Sel Darah Merah, Bayesian Classification, Algoritma Expectation-Maximization(EM)

OVERLAPPED RED BLOOD CELL IDENTIFICATION OF BLOOD MICROSCOPIC IMAGE USING UNSUPERVISED BAYESIAN CLASSIFICATION

Student's Name	**:**	**RM Reza Rizky Pratama**
Student's ID	**:**	**5110 100 181**
Department	**:**	**Informatics, FTIf-ITS**
First Advisor	**:**	**Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.**
Second Advisor	**:**	**Diana Purwitasari, S.Kom., M.Sc.**

Abstract

Number of cells can be used to determine disease's type like anemia caused by a lack of red blood cells or leukemia caused by the excess of white blood cells. When red blood cells are extracted automatically from microscopic images, the presence of stacked cells could lead to inaccuracies in counting the number of cells. This final project identified stacked cells from blood microscopic images for red blood cell count with unsupervised Bayesian classification.

Cluster model is used to separate stacked cells. Expectation-Maximization algorithm was used to approximate the optimal values for input parameters in Bayesian classifier. Cluster validity index is used to estimate the amount of overlapped red blood cells.

Experiments compared the values of red blood cell count from the implemented system with the result of manual counting. The error values of undersegmented and oversegmented indicators show red blood cells which are not precisely overlapped. The experiment results displayed some satisfactory results with the accuracy of 87,94% in addition to undersegmented error 3,82% and oversegmented error 8,24%.

Keywords: Overlapped Cell, Red Blood Cell, Bayesian Classification, Expectation-Maximization Algorithm (EM).

KATA PENGANTAR

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

Alhamdulillahirabbil'alamin, segala puji bagi Allah SWT, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul "Identifikasi Sel Darah Merah Bertumpuk Menggunakan *Unsupervised Bayesian Classification* pada Citra Mikroskopik Sel Darah**".**

Pengerjaan Tugas Akhir ini menjadi sebuah sarana bagi penulis untuk memperdalam ilmu pengetahuan dan mengimplementasikan apa yang didapatkan penulis selama menempuh perkuliahan di Teknik Informatika ITS.

Terselesaikannya buku Tugas Akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak. Sehingga pada kesempatan ini penulis mengucapkan terima kasih kepada:

1. Allah SWT atas limpahan rahmat dan karunia-Nya sehingga penulis diberikan kemampuan akal pikir yang mampu menghasilkan buah karya ini.
2. Kedua orang tua penulis, Bapak Dita Mahar Rochman dan Ibu Djumharini yang selalu memberikan motivasi dan semangat pada saat pengerjaan Tugas Akhir.
3. Ibu Dr. Eng. Chastine Fatichah, S.Kom., M.Kom. selaku dosen pembimbing 1 yang selalu sabar dalam menghadapi penulis dan memberikan bimbingan dan bantuan kepada penulis.
4. Ibu Diana Purwitasari selaku dosen pembimbing 2 yang dengan sabar memberikan masukan dan semangat pada penulis.
5. Teman – teman dari tim TFX yang memberikan dukungan moral serta selalu menyemangati penulis.
6. Seluruh teman Teknik Informatika ITS angkatan 2010.

7. Seluruh teman kelas Edhoc Teknik Informatika ITS angkatan 2010.
8. Seluruh pihak yang belum sempat disebutkan satu per satu yang telah membantu terselesaikannya Tugas Akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan. Sehingga dengan kerendahan hati, penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depan.

Surabaya, Juli 2015

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar Tugas Akhir yang meliputi latar belakang, tujuan, rumusan dan batasan permasalahan, metodologi pembuatan tugas akhir, dan sistematika penulisan. Latar belakang berisi mengenai hal-hal yang melatarbelakangi pemilihan judul Tugas Akhir. Rumusan masalah memuat hal-hal yang harus diselesaikan. Batasan masalah berisi batasan-batasan yang melingkupi pembuatan Tugas Akhir ini. Tujuan berisi tujuan dari pembuatan Tugas Akhir ini. Metodologi membahas mengenai metode pengerjaan Tugas Akhir. Sistematika penulisan membahas sistematika penulisan buku Tugas Akhir sebagai laporan dari pengerjaan Tugas Akhir. Penjelasan tentang hal-hal tersebut diharapkan dapat memberikan gambaran umum mengenai permasalahan sehingga penyelesaian masalah dapat dipahami dengan baik.

1.1 Latar Belakang

Setiap sel yang ada dalam tubuh manusia memegang peranan penting dalam kesehatan manusia. Jumlah sel merupakan salah satu acuan di dalam dunia kedokteran terhadap jenis penyakit, seperti anemia yang disebabkan kurangnya sel darah merah, leukimia yang disebabkan lebihnya jumlah sel darah putih, dan lain-lain. Salah satu sel yang memegang peranan penting di kehidupan adalah sel darah merah.

Penghitungan sel darah merah dapat dilakukan secara manual dengan menggunakan *hemocytometer* dan mikroskop, dan juga secara otomatis dengan menggunakan mesin *hematology analyzer* agar mempermudah penghitungan dan mempercepat proses penghitungan.

Dalam proses penghitungan sel darah merah secara otomatis, salah satu masalah yang ada adalah terdapatnya sel darah merah bertumpuk yang dapat mengakibatkan ketidakakuratan

penghitungan jumlah sel sehingga dapat menimbulkan kesalahan diagnosis yang berkaitan dengan jumlah sel darah merah. Telah banyak metode ataupun penelitian yang dikembangkan untuk mengatasi masalah sel yang bertumpuk, mulai dari permodelan sel menjadi *cluster* [1], penghitungan sel bertumpuk dengan fitur geometri [2] dan sebagainya.

Pada Tugas Akhir ini, identifikasi sel darah merah bertumpuk menggunakan *unsupervised Bayesian classification* pada citra mikroskopik sel darah, pemisahan sel yang bertumpuk akan dimodelkan sebagai suatu *cluster*, kemudian digunakan algoritma *EM (Expectation - Maximization)* untuk mendapatkan hasil parameter yang optimal sebagai nilai masukan *Bayesian classifier*, untuk mengamati *cluster* yang akan dibentuk digunakan *Bayesian classifier* [1]. Kemudian, *cluster validity index* digunakan untuk mengestimasi jumlah sel yang bertumpuk [3]. Diharapkan dengan metode ini dapat mengidentifikasi sel darah merah yang bertumpuk secara optimal.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini dapat dipaparkan sebagai berikut.

1. Bagaimana cara mengidentifikasi sel darah merah bertumpuk dari *input* citra sel darah merah?
2. Bagaimana cara menampilkan citra pemisahan sel darah merah bertumpuk?
3. Bagaimana cara melakukan penghitungan akurasi jumlah sel darah merah bertumpuk dari *input* citra sel darah merah?

1.3 Batasan Masalah

Tugas Akhir ini mempunyai banyak permasalahan, tetapi tidak semua permasalahan tersebut akan terselesaikan karena terdapat keterbatasan sumber daya. Maka dari itu, dibuat batasan-batasan permasalahan yang ada, antara lain batasannya sebagai berikut:

1. Implementasi menggunakan MATLAB 8.1 R2013a.
2. *Input* citra yang digunakan berasal dari citra mikroskopik sel darah.

1.4 Tujuan dan Manfaat

Tujuan pembuatan Tugas Akhir ini adalah untuk memudahkan penghitungan jumlah sel darah merah agar lebih akurat dengan menampilkan hasil citra pemisahan sel darah merah bertumpuk.

Manfaat dari Tugas Akhir ini yaitu sebuah penerapan metode untuk identifikasi sel darah merah bertumpuk yang diharapkan dapat membantu proses penghitungan jumlah sel darah merah agar lebih akurat.

1.5 Metodologi

Pembuatan Tugas Akhir ini terbagi menjadi beberapa tahap, adapun tahapan-tahapan yang dilakukan dalam pengerjaan Tugas Akhir ini adalah sebagai berikut:

1. Penyusunan Proposal Tugas Akhir

   Tahap awal pengerjaan Tugas Akhir adalah menyusun proposal Tugas Akhir. Gagasan yang diajukan pada proposal Tugas Akhir ini membantu mengidentifikasi sel darah merah bertumpuk menggunakan *unsupervised Bayesian classification* pada citra mikroskopik sel darah.
2. Studi literatur

   Pada tahap ini dilakukan pencarian informasi dan studi literatur yang diperlukan untuk pengumpulan data dan desain sistem yang akan dibuat. Referensi tersebut berisikan tentang:

   a. Konsep mengenai *Expectation–Maximization.*
   b. Konsep mengenai *Gaussian mixture models.*
   c. Konsep mengenai *Bayesian classifier.*

   Referensi tersebut dapat dicari dari buku, jurnal, artikel, laporan penelitian, dan situs-situs di internet. Studi literatur

ini adalah terkoleksinya referensi yang relevan dengan perumusan masalah. Tujuannya untuk memperkuat permasalahan serta sebagai dasar teori dalam melakukan studi dan juga menjadi dasar untuk melakukan identifikasi sel darah merah bertumpuk menggunakan *unsupervised Bayesian classification* pada citra mikroskopik sel darah.

3. Analisa dan implementasi sistem

   Pada tahap ini dilakukan analisa dan implementasi dari algoritma yang digunakan untuk membangun sistem tersebut. Untuk membangun algoritma yang telah dirancang sebelumnya, diimplementasikan dengan menggunakan Matlab.

4. Pengujian dan evaluasi

   Pada tahap ini dilakukan uji coba dengan menggunakan 10 citra mikroskopik sel darah untuk mencoba jalannya aplikasi apakah telah sesuai dengan rancangan dan desain implementasi yang dibuat. Pengujian hasil dilakukan dengan membandingkan hasil dari sistem secara otomatis dengan hasil penghitungan sel darah merah secara manual. Penghitungan akurasi dilakukan dengan 2 cara yaitu:

   1. Menghitung akurasi segmentasi dari citra hasil yang didapat dibandingkan dengan citra *ground truth*.
   2. Menghitung akurasi sel darah merah yang bertumpuk dengan mempertimbangkan *correctly segmented*, *undersegmented* dan *oversegmented* dari hasil yang didapat.

5. Penyusunan buku Tugas Akhir

   Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat.

1.6 Sistematika Penulisan Laporan Tugas Akhir

Sistematika penulisan laporan Tugas Akhir adalah sebagai berikut:

1. BAB I. PENDAHULUAN

   Bab ini berisi penjelasan mengenai latar belakang masalah, tujuan, dan manfaat dari pembuatan Tugas Akhir. Selain itu juga turut disertakan pula rumusan permasalahan, batasan permasalahan, dan sistematika penulisan pada bab ini.

2. BAB II. TINJAUAN PUSTAKA

   Bab ini membahas beberapa teori penunjang yang berhubungan dengan pokok pembahasan dan mendasari pembuatan Tugas Akhir ini, diantaranya : *Expectation–Maximization*, segmentasi citra, *Bayesian classification*, *Gaussian mixture models*, *cluster validity index*, dan sebagainya.

3. BAB III. PERANCANGAN PERANGKAT LUNAK

   Bab ini membahas mengenai perancangan perangkat lunak yang terdiri dari perancangan data dan perancangan sistem. Perancangan data berguna untuk menyiapkan dataset sebagai data masukan. Perancangan sistem adalah perancangan metode yang akan diimplementasikan.

4. BAB IV. Implementasi

   Bab ini berisi tentang penjelasan dan pembahasan mengenai implementasi program dari rancangan metode yang dijabarkan pada Bab III yang telah dilakukan.

5. BAB V. Pengujian dan Evaluasi

   Bab ini berisi penjelasan mengenai data hasil percobaan dan pembahasan mengenai hasil percobaan yang telah dilakukan.

6. BAB VI. Kesimpulan dan Saran

   Bab ini berupa hasil penelitian yang menjawab permasalahan atau yang berupa konsep, program, dan rancangan. Selain itu pada bab ini diberikan saran-saran yang berisi hal-hal yang berhubungan dengan penelitian

yang telah dilakukan untuk dikembangkan lebih lanjut atau berisi masalah-masalah yang dialami pada proses pengerjaan Tugas Akhir.

7. Daftar Pustaka

   Bab ini berisi daftar pustaka yang dijadikan literatur dalam pengerjaan Tugas Akhir.

8. Lampiran Tugas Akhir

   Dalam lampiran terdapat hasil gambar uji coba dan tabel-tabel data hasil uji coba.

BAB II
TINJAUAN PUSTAKA

Bab ini membahas tinjauan pustaka yang digunakan dalam penyelesaian masalah pada Tugas Akhir. Tinjauan Pustaka yang akan dijelaskan terbagi dalam beberapa subbab, yaitu citra digital, segmentasi citra, *Gaussian mixture models*, *Bayesian classification*, *distance transforrm*, *regional maxima*, *algortima nearest neighbour*, *algoritma Expectation-Maximization*, *cluster validiy index*. Materi-materi tersebut masing-masing akan dijelaskan dalam subbab tersendiri.

2.1 Citra Digital

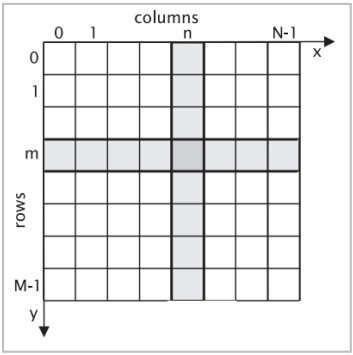
Citra digital merupakan representasi numerik dari citra dua dimensi. Citra dapat berupa foto mahluk hidup, pemandangan, benda mati, mikroskopik, dan sebagainya. Citra digital dapat dibuat menggnakan alat penghasil citra seperti kamera digital, smesin pengukur, mikroskop elektron dan sebagainya.

Suatu citra dapat didefinisikan sebagai fungsi dua dimensi yaitu $f(x, y)$, di mana x dan y merupakan koordinat spasial, dan amplitudo dari f pada pasangan koordinat (x, y) manapun, disebut sebagai intensitas atau tingkat keabuan dari citra di koordinat tersebut. Jika nilai $x, y,$ dan amplitudo dari fungsi f bersifat diskrit dan terbatas, maka dapat disebut sebagai citra digital [4].

Citra digital terdiri dari beberapa elemen dan setiap elemen tersebut memiliki nilai dan lokasi tertentu. Elemen gambar dapat disebut sebagai elemen citra atau piksel. Istilah piksel inilah yang sering dipakai untuk menyatakan elemen dari citra digital.

Letak piksel pada citra dua dimensi umumnya direpresentasikan dalam bentuk matriks. Representasi citra digital ditunjukkan oleh Gambar 2.1. Dari gambar tersebut dapat dilihat bahwa representasi citra digital dua dimensi ditampilkan dalam bentuk susunan kotak-kotak, yang biasa disebut piksel. Indeks *m* menyatakan posisi baris, dan indeks *n* menyatakan posisi kolom.

Jika citra digital terdiri dari $M \times N$ piksel, maka akan direpresentasikan dalam bentuk matriks berukuran $M \times N$, di mana indeks m memiliki rentang antara 0 hingga *M-1* dan indeks n memiliki rentang antara 0 hingga *N-1*. *M* menyatakan jumlah baris dan *N* menyatakan jumlah kolom. Sesuai dengan notasi matriks, sumbu vertikal (sumbu *y*) berjalan dari atas ke bawah dan sumbu horisontal (sumbu *x*) berjalan dari kiri ke kanan.



Gambar 2.1 Representasi Citra Digital

2.2 Segmentasi Citra

Segmentasi citra adalah suatu proses pengolahan citra yang membagi suatu citra menjadi beberapa objek. Dalam prosesnya dilakukan pembagian dari citra menjadi objek yang homogen / sama berdasarkan kriteria keserupaan yang tertentu antara tingkat keabuan suatu piksel dengan tingkat keabuan piksel-piksel tetangganya. Beberapa contoh dari segmentasi citra adalah *thresholding*, deteksi tepi, deteksi titik, deteksi garis dan sebagainya. Hasil dari proses segmentasi citra sering digunakan untuk melakukan proses tingkat tinggi lebih lanjut yang dapat dilakukan terhadap citra tersebut, misalnya proses klasifikasi citra dan proses identifikasi objek.

2.3 Gaussian Mixture Models

Gaussian mixture models adalah sebuah metode penganalisaan data atau data mining. Metode ini mengelompokkan data-data di dalam suatu *dataset* menjadi kelompok-kelompok data yang sebelumnya tidak terdefinisi dan berdasarkan dari distribusi *Gauss* atau distribusi normal. Distribusi normal adalah kurva yang mempunyai dua parameter, parameter pertama adalah $\boldsymbol{\mu}$ atau *mean* dan parameter kedua adalah $\boldsymbol{\sigma}$ atau standar deviasi. Kumpulan distribusi normal biasa disebut dengan *normal probability density function* atau *normal pdf*.

Multivariate Gaussian mixture models adalah variasi dari *Gaussian mixture models* dimana terdapat dua atau lebih variabel yang dapat mempengaruhi suatu hasil distribusi, salah satu contoh nya seperti analisis pengaruh kualitas produk dan harga produk terhadap tingkat penjualan suatu produk.

Parameter dari GMM diantaranya adalah rata-rata suatu sampel ($\boldsymbol{\mu}$), kovarians ($\boldsymbol{\Sigma}$) dan probabilitas (т), namun parameter tersebut nilainya bergantung pada sampel set data, data tersebut terdiri dari variabel independen dan variabel laten [5].

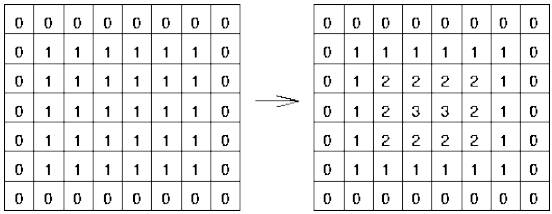
2.4 Bayesian Classification

Bayesian classification adalah klasifikasi statistik yang dapat memprediksi kelas suatu anggota probabilitas. Untuk klasifikasi *Bayes* sederhana yang lebih dikenal sebagai *naïve Bayesian classifier* dapat diasumsikan bahwa efek dari suatu nilai atribut sebuah kelas yang diberikan adalah bebas dari atribut-atribut lain. Asumsi ini disebut *class conditional independence* yang dibuat untuk memudahkan penghitungan [6].

Hasil dari *Bayes classifier* yang didapatkan nanti akan digunakan sebagai penentu suatu sampel data $\mathbf{x}_i$ yang akan dikelompokan pada *cluster j* dilihat dari nilai probabilitas yang terbesar diantara nilai probabilitas sampel data $\mathbf{x}_i$ dari 1 hingga m.

2.5 Distance Transform

Distance transform adalah salah satu proses transformasi citra biner menjadi citra *distance*, *distance transform* mengubah setiap piksel *foreground* pada image menjadi piksel *distance transform*, piksel *distance transform* didapatkan dari mengukur jarak minimal suatu piksel *foreground* tersebut dengan piksel *background* dengan menggunakan beberapa metode, salah satunya dengan *Euclidean distance*.



Gambar 2.2 Perubahan citra biner menjadi citra *distance*

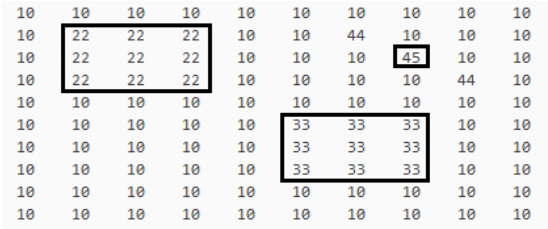
Pada Gambar 2.2 ditampilkan perubahan citra biner menjadi citra *distance*, semakin jauh jarak piksel *foreground* dari piksel *background*, maka semakin besar nilai *distance* yang dimiliki piksel *foreground* tersebut [7].

Beberapa penggunaan *distance transform* pada proses citra digital diataranya *blurring effects*, *skeletonizing*, *pathfinding* dan sebagainya. Pada Tugas Akhir ini, metode *distance transform* digunakan sebagai pembentukan *Gaussian mixture model* yang kemudian diproses untuk mendapatkan nilai *regional maxima* yang akan dijadikan inisialisasi *cluster* awal.

2.6 Regional Maxima

Regional maxima adalah komponen piksel yang saling terhubung dan memliki nilai piksel yang lebih besar dibandingkan nilai piksel di sekitarnya, penggunaan *regional maxima* adalah

untuk mendapatkan nilai puncak *maxima* pada suatu region dalam citra [8].



Gambar 2.3 *Regional maxima* dari matriks A 10 x 10

Pada Gambar 2.3 ditunjukan representasi dari *regional maxima* pada matriks A 10 x 10, kumpulan piksel dengan nilai 22 menjadi *regional maxima* karena disekelilingnya terdapat piksel yang lebih kecil nilainya yaitu 10. Pada kumpulan piksel dengan nilai 33 juga memiliki karakteristik yang sama dengan piksel 22. Pada piksel dengan nilai 45, piksel sekelilingnya adalah 44 dan 10, sehingga piksel dengan nilai 45 menjadi titik *regional maxima*, sedangkan piksel dengan nilai 44 tidak menjadi *regional maxima* karena terdapat piksel dengan nilai 45 yang nilainya lebih besar.

Penggunaan *regional maxima* pada Tugas Akhir ini adalah sebagai metode untuk menentukan jumlah *cluster* awal suatu objek sel darah merah. Piksel *regional maxima* yang membentuk kumpulan seperti piksel 33 pada Gambar 2.3 akan dihitung menjadi satu anggota *cluster*, sehingga pada matriks A 10 x 10 memiliki nilai anggota *cluster* sebesar tiga.

Jumlah dari anggota *cluster regional maxima* yang ditemukan seperti pada contoh sebelumnya akan menjadi jumlah *cluster* awal yang akan dipakai pada proses selanjutnya.

2.7 Nearest Neighbour Algorithm

Nearest neighbour algorithm adalah salah satu algoritma *clustering* yang menentukan suatu titik sembarang pada suatu kumpulan sampel data sebagai titik tengah *cluster* (*s*), penentuan

jumlah *cluster* ditentukan dari inisialisasi awal, jika banyak *cluster* yang ditentukan adalah *m*, maka terdapat *m* titik tengah *cluster* $(s_1, s_2, \dots s_m)$. Kemudian dari titik tengah *cluster* tersebut dilakukan pencarian jarak terpendek suatu piksel **x** terhadap titik tengah *cluster* tersebut [6].

$$X = \{x \mid i = \arg\min_k \|\mathrm{x} - s_k\|^2, \mathrm{x} \in \mathrm{X}, 1 \leq k \leq m\} \quad (2.1)$$

Pada Persamaan (2.1), hasil jarak terpendek dari titik tengah *cluster* akan digunakan sebagai penentuan titik sampel piksel **x** dikelompokan ke dalam *cluster k*, namun hasil dari *nearest neighbour algorithm* tidak terlalu optimal dalam proses *clustering*, karena dilakukan inisialisasi titik tengah *cluster* secara *random*.

2.8 Expectation–Maximization Algorithm

Expectation-maximization (EM) *algorithm* adalah sebuah algoritma untuk mengestimasi suatu nilai di dalam model dari suatu variabel laten yang tidak diketahui, algoritma EM merupakan iterasi yang melakukan penggabungan dari dua proses yaitu proses *expectation* (E) fungsi untuk mengekspetasi nilai parameter yang dianggap memiliki nilai logaritma kemungkinan (*log-likelihood*) yang tinggi dan *maximization* (M) di mana parameter *log-likelihood* yang didapatkan dari proses E dapat dimaksimalkan. Kemudian parameter yang dihasilkan dari proses M dapat digunakan untuk menentukan distribusi dari variabel laten untuk proses E selanjutnya [9].

Implementasi algoritma EM pada GMM bergantung pada sampel variable independen dan variable laten. Variabel independen dari *Multivariate* GMM dimisalkan **X**, dimana **X** = $(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$ yaitu sampel berjumlah n yang diperoleh dari *Multivariate* GMM. Nilai dari $\mathbf{x}_i$ adalah $\mathbf{x}_i = [x\ y]$, yaitu posisi atau letak koordinat sampel data i pada suatu GMM. Variabel laten dimisalkan Z, dimana Z = $(z_1, z_2, \dots, z_m)$. Pada GMM, variabel Z menunjukan *cluster* yang ditentukan sebanyak *m*.

Tujuan utama dari algortima EM adalah mengestimasi nilai *likelihood* dengan proses E dan mengoptimalkan nilai *likelihood* dengan proses M, selanjutnya berdasarkan nilai optimal yang didapatkan pada proses M, dilakukan distribusi variable laten terhadap variabel independen, yaitu pengelompokan *cluster*.

Pada algoritma EM, dilakukan proses untuk mencari *unknown parameter* (parameter yang tidak diketahui) dan *missing value* (nilai yang hilang). Parameter yang tidak diketahui dilambangkan dengan $\boldsymbol{\theta}$, seperti yang ditunjukan pada Persamaan (2.2).

$$\boldsymbol{\theta} = (\tau, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2, \dots, \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m) \tag{2.2}$$

Variabel τ adalah *prior probability* yang didapat dari inisialisasi pengelompokan awal dari *cluster* z, yang dapat dilambangkan $P(Z_i = 1) = \tau_1$, $P(Z_i = 2) = \tau_2$, … $P(Z_i = m) = \tau_m$ dimana $\tau_1 + \tau_2 + \dots \tau_m = 1$.

Variabel τ_i menunjukan probabilitas sampel Z_i pada keseluruhan sampel data dimana *i* menunjukan *cluster* ke - *i*. Variabel $\boldsymbol{\mu}$ adalah rata-rata dari sampel **x** yang merupakan koordinat suatu sampel data, maka $\boldsymbol{\mu}_i$ berupa vector yang berisi $[\bar{x}_i\ \bar{y}_i]$ dimana $\bar{x}_i$ adalah rata-rata dari sampel data *x* yang termasuk pada *cluster* z_i dan $\bar{y}_i$ adalah rata-rata dari sampel data *y* yang termasuk pada *cluster* z_i.

Variabel $\boldsymbol{\Sigma}$ adalah matriks kovarian dari sampel **x**, untuk mendapatkan matriks kovarian, dilakukan penghitungan dengan sampel $\mathbf{x}_i$ dan $\boldsymbol{\mu}_i$. Karena sampel $\mathbf{x}_i$ merupakan kumpulan dari sampel **x** pada *cluster* ke – *i*, dan isi dari sampel $\mathbf{x}_i$ berbentuk matriks $k \times 2$ dimana *k* adalah jumlah variabel sampel **x** yang termasuk pada *cluster* ke – *i*. Maka bentuk dari sampel $\mathbf{x}_i$ ditampilkan seperti pada Persamaan (2.3).

$$\mathbf{x}_i = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ x_k & y_k \end{bmatrix} \tag{2.3}$$

Matriks $k \times 2$ tersebut akan dipisah menjadi 2 vektor yaitu vector X berukuran $1 \times k$, $X = [x_1, x_2, \dots, x_k]$ dan vector Y berukuran $1 \times k$, $Y = [y_1, y_2, \dots, y_k]$, kemudian dilakukan penghitungan kovarian yang ditunjukan pada Persamaan (2.4).

$$\sigma_{xy} = \frac{1}{k} \sum_{i=1}^{k} (x_i - \bar{\text{x}})\,(y_i - \bar{\text{y}})^T \tag{2.4}$$

Matriks kovarian akan terbentuk dengan ukuran $m \times m$, dimana m adalah jumlah *cluster* yang telah ditentukan sebelumnya, maka matriks kovarian yang akan dibentuk seperti pada Persamaan (2.5).

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_{x^2} & \sigma_{xy} \\ \sigma_{xy} & \sigma_{y^2} \end{bmatrix} \tag{2.5}$$

Nilai *likelihood* didapatkan dari hasil penghitungan *prior probability* dan *class conditional probability*, dimana prior probability dilambangkan dengan τ dan class conditional probability dilambangkan dengan $f(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$.

Pada GMM, *class conditional probability* adalah *probability density function*, untuk menghitung *class conditional probability* dibutuhkan sampel data $\mathbf{x}$, rata-rata data $\boldsymbol{\mu}$ dan matriks kovarians $\boldsymbol{\Sigma}$, seperti pada Persamaan (2.6).

$$f(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{2\pi|\boldsymbol{\Sigma}|}} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})\right] \tag{2.6}$$

Kemudian nilai dari $f(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ dikalikan dengan τ dan dijumlah sebanyak m kali, yaitu bergantung dengan jumlah *cluster* nya, maka penghitungan nilai *likelihood* ditunjukan pada Persamaan (2.7).

$$L(\boldsymbol{\theta}\,; \mathbf{x}) = \prod_{i}^{n} \sum_{j=1}^{m} \tau_j f(\mathbf{x_i}; \boldsymbol{\mu_j}, \boldsymbol{\Sigma_j}) \tag{2.7}$$

Dimana n adalah jumlah sampel dari data **X** dan m adalah jumlah *cluster* yang ditentukan sebelumnya, sehingga didapatkan nilai *likelihood*.

Pada proses E dilakukan pembentukan nilai dari *likelihood* dan probabilitas anggota (T), pembentukan probabilitas anggota menggunakan teori *bayes classifier* yang di proporsikan menggunakan *prior probability*, seperti pada Persamaan (2.8).

$$T_{j,i}^{(t)} \coloneqq P\left(Z_i = j \middle| X_i = \mathbf{x}_i; \boldsymbol{\theta}^{(t)}\right) = \frac{\tau_j^{(t)} f\left(\mathbf{x_i}, \boldsymbol{\mu_j}^{(t)}, \boldsymbol{\Sigma_j}^{(t)}\right)}{\tau_1^{(t)} f\left(\mathbf{x_1}, \boldsymbol{\mu_1}^{(t)}, \boldsymbol{\Sigma_1}^{(t)}\right) + \tau_2^{(t)} f\left(\mathbf{x_i}, \boldsymbol{\mu_2}^{(t)}, \boldsymbol{\Sigma_2}^{(t)}\right) + \ldots + \tau_m^{(t)} f\left(\mathbf{x_i}, \boldsymbol{\mu_m}^{(t)}, \boldsymbol{\Sigma_m}^{(t)}\right)} \tag{2.8}$$

Dimana (t) menunjukan kondisi sekarang atau kondisi awal saat proses E dilakukan, berikutnya setelah proses M dijalankan dan kembali pada proses E, maka (t) menjadi $(t+1)$. T akan berbentuk sebuah matriks dengan ukuran $m \times n$ yang berbanding dengan jumlah *cluster* × jumlah sampel.

Pada proses M dilakukan kembali penghitungan mencari nilai prior probability (τ), rata-rata sampel ($\boldsymbol{\mu}$) dan matriks kovarians sampel ($\boldsymbol{\Sigma}$) berdasarkan nilai T yang didapatkan pada proses E. Penghitungan $\tau^{(t+1)}$yang baru dilakukan dengan melibatkan probabilitas anggota (T) seperti pada Persamaan (2.9).

$$\mathrm{T}_{j,i}^{(t+1)} = \frac{\sum_{i=1}^{n} T_{j,i}^{(t)}}{\sum_{i=1}^{n}(T_{1,i}^{(t)} + T_{2,i}^{(t)} + \cdots T_{m,i}^{(t)})} \quad (2.9)$$

Jumlah dari prior probability yang baru melibatkan probabilitas anggota pada *cluster j* dibagi dengan jumlah dari semua probabilitas anggota dari *cluster* 1 hingga *cluster* m.

Pembentukan rata-rata sampel ($\boldsymbol{\mu}^{(t+1)}$) yang ditunjukan pada Persamaan (2.10) dan matriks kovarians ($\boldsymbol{\Sigma}^{(t+1)}$) yang ditunjukan pada Persamaan (2.11) juga dipengaruhi oleh probabilitas anggota (T) dengan rumusan untuk masing-masing variabel.

$$\boldsymbol{\mu}_j^{(t+1)} = \frac{\sum_{i=1}^{n} T_{j,i}^{(t)} \mathbf{x}_i}{\sum_{i=1}^{n} T_{j,i}^{(t)}} \quad (2.10)$$

$$\boldsymbol{\Sigma}_j^{(t+1)} = \frac{\sum_{i=1}^{n} T_{j,i}^{(t)} (\mathbf{x}_i - \boldsymbol{\mu}_j^{(t+1)})(\mathbf{x}_i - \boldsymbol{\mu}_j^{(t+1)})^T}{\sum_{i=1}^{n} T_{j,i}^{(t)}} \quad (2.11)$$

Kemudian dari hasil $\boldsymbol{\mu}^{(t+1)}$ dan $\boldsymbol{\Sigma}^{(t+1)}$ yang baru dilakukan pencarian *likehood* yang baru $L(\boldsymbol{\theta}\,;\boldsymbol{x})^{t+1}$ yang akan dibandingkan dengan *likelihood* yang lama.

Perulangan dari proses algoritma EM akan diakhiri jika nilai *log-likelihood* yang baru lebih dari atau sama dengan *log-likelihood* yang lama seperti pada Persamaan (2.12).

$$\log L(\boldsymbol{\theta}\,;\boldsymbol{x})^{t} \leq \log L(\boldsymbol{\theta}\,;\boldsymbol{x})^{t+1} \quad (2.12)$$

2.9 Cluster Validity Index

Cluster validity index adalah tingkat kevalidan suatu *cluster* yang ada terhadap suatu sampel data. Penggunaan *cluster validiy*

index sangat penting untuk menentukan apakah jumlah *cluster* yang ditentukan telah optimal atau belum, terdapat beberapa cara untuk menentukan *cluster validity index*, salah satunya adalah dengan menggunakan penggabungan antara fungsi *separation* dan *compactness* [3].

$$V(m) = \frac{\varepsilon(m)}{\varphi(m)} \tag{2.13}$$

Pada Persamaan (2.13), $V(m)$ adalah *cluster validity index* yang merupakan hasil bagi antara *separation* dan *compactness*, nilai dari *separation* adalah hasil dari jumlah diagonal elemen pada *weighted between scatter matrix* yang ditunjukan pada Persamaan (2.14).

$$\epsilon = trace(S) \tag{2.14}$$

Variabel S menunjukan *between scatter matrix* yang mempunyai bobot dan dapat didefinisikan pada Persamaan (2.15).

$$S = \sum_{i=1}^{m} \sum_{k=1}^{n} p_{ki} \, (\mathrm{m}_i - \bar{\mathrm{m}})(\mathrm{m}_i - \bar{\mathrm{m}})^T \tag{2.15}$$

Dimana variabel m adalah jumlah *cluster*, n adalah jumlah piksel sampel data, p_{ki} adalah *posterior probability*, m_i adalah titik tengah *cluster* ke-i, dan $\bar{\mathrm{m}}$ adalah titik tengah sampel data. Kemudian nilai dari *compactness* adalah hasil dari jumlah diagonal elemen pada *normalized covariance matrix* seperti pada Persamaan (2.16).

$$\varphi = \sum_{i=1}^{m} trace(C) \tag{2.16}$$

Variabel C menunjukan *normalized covariance matrix* yang mempunyai bobot dan didefinisikan pada Persamaan (2.17).

$$C = \frac{\sum_{k=1}^{n} p_{ki}(\mathrm{x}_k - \mathrm{m}_i)(\mathrm{x}_k - \mathrm{m}_i)^T}{\sum_{k=1}^{n} p_{ki}} \quad (2.17)$$

Dimana variabel m adalah jumlah *cluster*, n adalah jumlah piksel sampel data, p_{ki} adalah *posterior probability*, x_k adalah titik sampel piksel ke - k dan m_i adalah titik tengah *cluster* ke-i.

BAB III
PERANCANGAN PERANGKAT LUNAK

Bab ini menguraikan perancangan sistem dari identifIkasi sel darah merah bertumpuk menggunakan *unsupervised Bayesian classifier*. Perancangan yang diuraikan meliputi lingkungan perancangan perangkat lunak, perancangan data, dan perancangan serta *pseudocode* proses.

3.1 Lingkungan Perancangan Perangkat Lunak

Proses perancangan perangkat lunak Tugas Akhir ini membutuhkan lingkungan perancangan yang sesuai. Tahapan implementasi dan pengujian dilakukan dengan menggunakan perangkat keras dan perangkat lunak yang digunakan untuk proses identifikasi sel darah merah bertumpuk pada citra mikroskopis sel menggunakan *unsupervised Bayesian classification*.

Perangkat keras yang digunakan mempunyai spesifikasi sebagai berikut:

- Laptop ASUS A450L
  - Windows 8.1 64-bit.
  - Prosesor Intel(R) Core(TM) i5-4200U CPU @ 1.60GHz.
  - RAM : 4 GB.

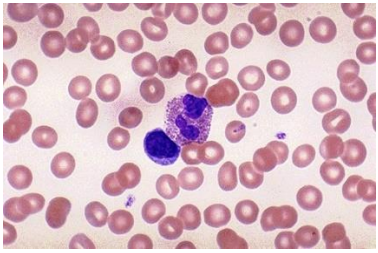
  Sedangkan perangkat lunak yang digunakan adalah:
- Windows 8.1 64-bit sebagai sistem operasi.
- MATLAB 8.1 R2013a sebagai aplikasi pendukung.

3.2 Perancangan Data

Perancangan data sangat diperlukan agar perangkat lunak dapat berjalan dengan baik dan benar. Data yang diperlukan dalam suatu perangkat lunak dibagi menjadi tiga, yaitu data masukan (*input*), data proses yaitu data hasil proses sistem, dan data keluaran (*output*) yaitu data hasil keluaran akhir dari sistem.

3.2.1 Data Masukan

Data masukan yang digunakan dalam sistem ini adalah kumpulan data citra mikroskopis sel darah. Kumpulan citra darah tersebut terdiri dari 10 buah citra mikrokopis sel darah. Tiap citra terdiri dari sel darah merah, sel darah putih, keping darah, dan *background*. Gambar 3.1 merupakan sampel yang akan digunakan sebagai masukan pada sistem.



Gambar 3.1 Sampel Citra Mikroskopis Sel Darah

3.2.2 Data Proses

Data proses adalah data yang dihasilkan pada tahapan *Preprocessing* dan Segmentasi sel darah merah. dan data ini digunakan untuk proses selanjutnya. Data proses yang digunakan meliputi nama data, tipe data, dan keterangannya. Data proses secara detail disajikan pada Tabel 1.

Tabel 1 Data pada pemrosesan identifikasi sel darah merah bertumpuk

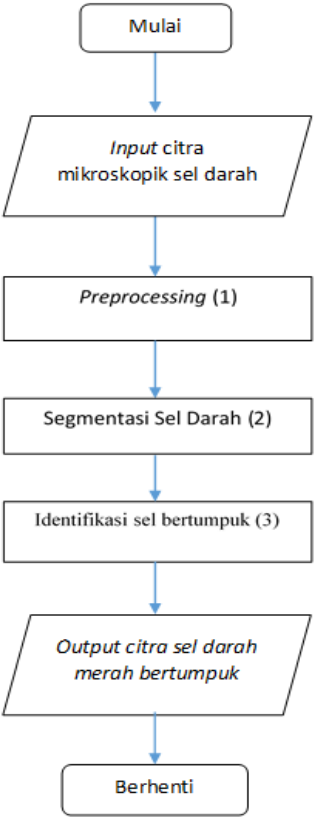
No.	Nama Data	Tipe Data	Keterangan
1	Citra biner sel darah	*binary*	Citra hasil *preprocessing* dari *input* citra RGB sel darah
2	Citra biner sel darah merah	*binary*	Citra hasil segmentasi dari *input* citra biner sel darah
3	Matriks sampel data objek sel darah merah	*binary*	Matriks hasil pemotongan tiap objek dari citra biner sel darah merah
4	Matriks *cluster* sampel data objek sel darah merah	*binary*	Matriks hasil dari proses *cluster validation*, dan intensitas dari matriks tersebut berupa label yang menunjukan anggota *cluster*, misal : piksel dengan intensitas '3' maka termasuk *cluster* ke 3
5	Citra visualisasi pemisahan sel darah merah yang bertumpuk	RGB	Citra hasil visualisasi pemisahan sel darah merah yang bertumpuk didapatkan dari Matriks *cluster*, citra tersebut merupakan hasil konversi dari *binary* label menjadi RGB

3.2.3 Data Keluaran

Data keluaran berupa citra hasil penambahan masukan citra RGB mikroskopis sel darah dan citra visualisasi pemisahan sel darah merah bertumpuk.

3.3 Perancangan serta *Pseudocode* proses

Bagian ini menjelaskan mengenai rancangan model metode dan proses pada bab sebelumnya dengan menggunakan *pseudocode* dan diagram alir. Bagan utama dari proses identifikasi sel darah merah bertumpuk ditunjukan pada Gambar 3.2.



Gambar 3.2 Diagram alir tahapan identifikasi sel darah merah bertumpuk pada citra sel darah merah

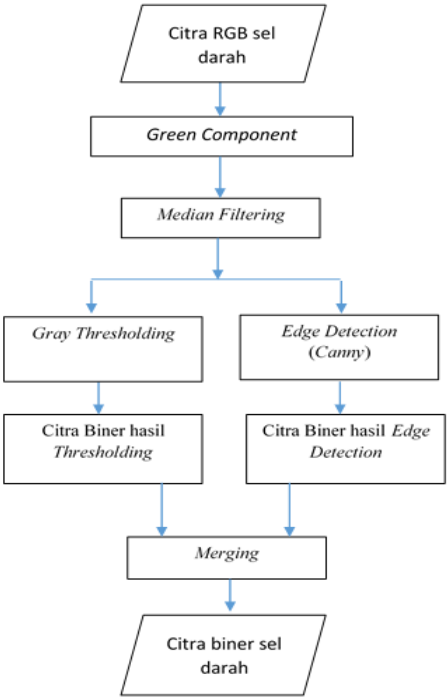
Pada diagram alir bagan utama terdapat tiga tahapan yang dilakukan, yaitu: *Preprocessing*, Segmentasi sel darah dan Identifikasi sel bertumpuk yang akan dibahas rinciannya pada subbab berikutnya.

3.3.1 Preprocessing

Tahap 1 yaitu *Preprocessing* adalah tahapan untuk mengubah input citra mikroskopik sel darah menjadi citra biner sel darah merah dilakukan sebanyak lima proses yaitu *Green Component*, *Median Filtering*, *Gray Thresholding*, *Edge detection (canny)*, dan *Merging*. Lima proses tersebut dilakukan pada *input* citra sel darah, hasil dari proses tersebut akan dilanjutkan pada proses selanjutnya. Alur dari *preprocessing* ditunjukan pada Gambar 3.3.

Pada Gambar 3.3, tahapan *Preprocessing* diawali dengan memproses citra RGB sel darah menjadi citra *green component* langkah ini dilakukan agar dapat menghilangkan sel darah putih pada proses selanjutnya. Kemudian dilakukan *filtering* terhadap *noise* dengan menggunakan *median filtering* yang dapat digunakan untuk mengurangi *noise* bintik - bintik atau disebut juga '*salt and pepper*' *noise* pada citra.

Gray thresholding digunakan untuk mengubah derajat keabuan pada citra yang akan digunakan unuk memisahkan obyek dan latar belakang citra. Jika derajat keabuan suatu piksel lebih besar dari batas *thresholding*, maka piksel tersebut dapat disebut objek dan jika derajat keabuan suatu piksel tersebut lebih kecil dari batas *thresholding*, maka piksel tersebut disebut latar belakang citra (*background*). Setelah itu citra *grayscale* tersebut dikonversi menjadi citra biner.



Gambar 3.3 Diagram alir tahapan *Preprocessing*

Citra biner yang dihasilkan nantinya kemungkinan akan memiliki bentuk lubang seperti cincin hal ini disebabkan karena saat pengambilan citra RGB tersebut terkena cahaya sehingga saat dilakukan proses pengubahan biner, daerah tersebut dianggap sebagai latar belakang citra / *background*.

Lubang dalam citra biner tersebut dapat didefinisikan sebagai sekumpulan piksel *background* yang dikelilingi dan tertutup oleh piksel *foreground*. Sehingga dapat menyebabkan terjadinya dua jenis lubang yang harus diperbaiki, yaitu:

a. Bentuk lubang tertutup seperti cincin pada objek sel
b. Bentuk lubang terbuka pada objek sel
c. Bentuk lubang terbuka yang bersentuhan dengan tepi

Perbaikan citra untuk lubang pertama dapat menggunakan operasi pengisian area *foreground* yang berlubang dengan menggunakan fungsi bawaan MATLAB yaitu *imfill*, untuk perbaikan lubang kedua dilakukan operasi *merging canny edge detection* dan citra biner. *Canny edge detection* mampu mendeteksi batas tepi citra, sehingga akan menutup tepi piksel yang terbuka. Setelah tepi piksel tertutup, langkah berikutnya adalah dilakukan proses operasi *imfill*.

Untuk objek yang terdapat di tepi citra yang memiliki masalah seperti pada kasus ke dua akan diatasi dengan merubah piksel yang terkena tepi dan mempunyai tetangga nilai piksel nya menjadi 1 sehingga akan tertutup, kemudian dilakukan proses *imfill*.

Pada Gambar 3.4 dan Gambar 3.5 merupakan *pseudocode* dari tahapan *Preprocessing*, hasil akhir dari proses ini adalah citra biner yang lebih maksimal tingkat ketepatan sel nya dibandingkan citra biner sebelumnya.

```
Masukan: citra RGB sel darah

1.  Konversi masukan Citra RGB menjadi green component
2.  Melakukan proses median filtering dari citra green
    component
3.  Membuat dua citra hasil dari median filtering dan
    dilakukan dua proses pada kedua citra tersebut, yaitu
    gray thresholding dan edge detection canny
4.  Hasil proses thresholding akan dirubah menjadi citra
    biner, kemudian dilakukan komplemen dari citra
    tersebut
```

Gambar 3.4 *Pseudocode* dari tahapan *Preprocessing* (Bagian 1)

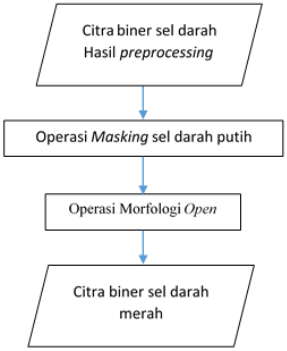
```
5.  Hasil proses dari edge detection canny adalah bentuk
    tepi dari sel dan merupakan citra biner
6.  Menggabungkan dua citra hasil dari dua proses diatas
    menjadi satu citra biner sel yang baru
7.  Dilakukan pengisian lubang dari hasil citra biner pada
    proses sebelumnya pada setiap sel

Keluaran: citra biner sel darah
```

Gambar 3.5 *Pseudocode* dari tahapan *Preprocessing* (Bagian 2)

3.3.2 Segmentasi Sel Darah Merah

Citra biner sel darah putih dan platelet merupakan *background* yang tidak dibutuhkan karena yang akan diproses pada tahapan identifikasi sel bertumpuk hanyalah citra biner sel darah merah. Oleh sebab itu citra biner sel darah putih harus dihilangkan. Alur dari proses pemisahan citra biner sel darah merah dengan *background* yang tidak diperlukan seperti terlihat pada Gambar 3.6.



Gambar 3.6 Diagram alir tahapan Segmentasi

Pada tahapan ini citra sel darah putih dihilangkan dengan operasi *masking*. Citra sel darah putih dapat dikenali karena memiliki ukuran yang relatif lebih besar dibandingkan sel darah merah tunggal, dan juga memiliki warna yang kontras yakni biru keunguan.

Sedangkan citra platelet memiliki ukuran yang relatif lebih kecil sehingga untuk menghilangkannya cukup dengan menggunakan operasi morfologi *open* bawaan MATLAB yaitu *bwareaopen*.

Gambar 3.7 merupakan *pseudocode* dari proses penghilangan sel darah putih pada citra biner.

```
Masukan: citra RGB sel darah,citra biner sel darah

1.  Melakukan deteksi objek sel darah putih berdasarkan
    warna objek dari citra RGB sel darah
2.  Mengambil hasil dari deteksi objek dengan menggunakan
    intensitas hasil piksel yang baru sehingga menjadi citra
    biner sel darah putih
3.  Dilakukan proses dilasi untuk memperlebar ukuran dari
    citra biner sel darah putih dan dilakukan pembersihan
    noise kecil
4.  Menghilangkan objek biner sel darah putih dari citra
    biner sel darah dengan melakukan proses pengurangan
    citra

Keluaran: citra biner sel darah merah
```

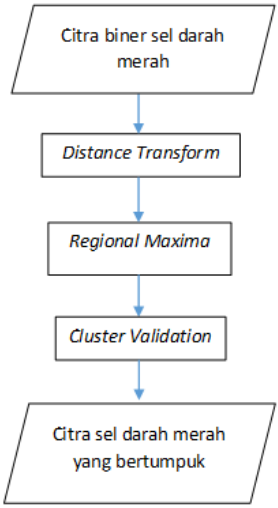
Gambar 3.7 *Pseudocode* dari segmentasi sel darah merah

3.3.3 Identifikasi Sel Bertumpuk

Identifikasi sel bertumpuk adalah tahapan dilakukannya identifikasi sel darah merah bertumpuk dari citra *regional maxima* yang didapat dari citra hasil *distance transform* dan citra biner dari tahapan sebelumnya. Terdapat 3 proses yaitu: *distance transform*, *regional maxima*, *cluster validation* seperti yang ditunjukan pada Gambar 3.8.

Hasil dari segmentasi citra sebelumnya menghasilkan citra biner sel darah merah yang telah bersih tanpa ada objek sel darah

putih ataupun platelet. Kemudian dari citra biner sel darah tersebut dilakukan proses *distance transform* untuk membuat *distance* image dan *regional maxima* yaitu proses pencarian titik *regional maxima* dari d*istance image* untuk mendapatkan nilai inisialisasi titk tengah *cluster* dan nilai jumlah *cluster* untuk setiap objek sel darah merah yang sedang di proses.



Gambar 3.8 Diagram alir tahapan Identifikasi sel bertumpuk

Pada proses *distance transform*, citra sel biner darah merah tersebut dirubah menjadi *distance image*, agar dapat dimodelkan menjadi distribusi *Gaussian*. *Distance image* nantinya akan dipakai sebagai masukan pada proses *regional maxima*.

Proses *regional maxima* sendiri digunakan untuk inisialisasi *cluster* awal dari *distance image* yang nanti akan diproses keoptimalannya pada proses *cluster validation*.

Pada Gambar 3.9 ditunjukan *pseudocode* dari proses *distance transform* dan *regional maxima* yang menghasilkan

keluaran titik tengah *cluster* setiap objek sel darah merah, dan jumlah *cluster* yang terbentuk setiap objek sel darah merah.

```
Masukan: citra biner sel darah merah

1.  Melakukan deteksi objek sel darah merah dan dilakukan
    pelabelan untuk setiap objek sel darah merah yang
    ditemukan
2.  Melakukan proses distance transform pada setiap objek
    sel darah merah yang ditemukan
3.  Melakukan proses pencarian regional maxima dari setiap
    objek hasil distance transform sel darah merah
4.  Melakukan pengelompokan dari hasil regional maxima yang
    ditemukan dan disimpan pada array of array variabel
    ovlpdistg
5.  Mencari nilai rata-rata posisi dari setiap variabel
    ovlpdistg sehingga menjadi inisialisasi titik tengah
    cluster pada setiap objek sel darah merah dan disimpan
    pada array variabel centinit
6.  Mencari jumlah anggota variabel centinit untuk setiap
    objek sel darah merah sebagai jumlah cluster awal yang
    disimpan pada variabel nb

Keluaran: variabel centinit,variabel nb
```

Gambar 3.9 *Pseudocode* dari proses *distance transform* dan *regional maxima*

Proses *cluster validation* dilakukan untuk menghitung jumlah sel darah merah bertumpuk dari *cluster* yang telah terbentuk. Pada *cluster validation* terdapat *cluster validity index* yang dipakai sebagai acuan jumlah *cluster* yang optimal dan dipakai [1].

Untuk setiap objek sel darah merah pada citra biner sel darah merah, dilakukan proses *cluster validation*. *Pseudocod*e dari identifikasi sel bertumpuk yang memproses keseluruhan citra biner sel darah merah yang dilakukan ditunjukan pada Gambar 3.10.

```
Masukan : seluruh objek sel darah merah (seed)

1.  Lakukan iterasi A sebanyak jumlah objek sel darah merah
    pada citra biner sel darah merah, sehingga terdapat
    objek sel darah merah yang dapat diakses dari indeks –
    i hingga jumlah objek sel darah merah. Matriks seed{i}
    menunjukan objek pada indeks – i.
2.  Lakukan inisialisasi posisi setiap objek seed{i}
3.  Lakukan proses clusterValidation
4.  Jika semua objek sudah diproses maka keluar dari
    iterasi A

Keluaran : objek seed yang telah ditentukan jumlah dan
bentuk cluster secara optimal
```

Gambar 3.10 Proses identifikasi sel bertumpuk seluruh objek

Pengerjaan dari proses *cluster validation* melibatkan algoritma EM dan *H-maxima transform* yang telah dijelaskan sebelumnya pada subbab 2.8. Pada Gambar 3.11 dan Gambar 3.12 ditampilkan *pseudocode* untuk *cluster validation*.

```
Masukan : sampel objek sel darah merah, nilai cardinalitas
regional maxima
1.  Inisialisasi jumlah seed m = # regional maxima
2.  Inisialisasi indeks cluster validity maksimal V_maks =
    −∞
3.  Inisialisasi nilai depth λ = 0
4.  Selama m > 1, lakukan langkah 5 - 15
5.  Ulangi langkah 6 – 7 hingga # regional maxima H_λ(I) ≥ m
6.  Perbarui nilai depth λ = λ + 1
7.  Evaluasi nilai transformasi H_λ(I)
8.  Hentikan perulangan bila nilai # regional maxima H_λ(I) ≥
    m
9.  Jika  # regional maxima H_λ(I) = m, lakukan langkah 10 –
    14
10. Konstruksi set R dari H_λ(I) dengan algoritma nearest
    neighbour
11. Lakukan algoritma EM
12. Estimasi indeks cluster validity V(m)
```

Gambar 3.11 *Pseudocode* dari *cluster validation* (Bagian 1)

```
13. Jika V(m) > V_maks, maka lakukan langkah 14
14. Perbarui nilai V_maks = V(m), m̂ = m, R̂ = R
15. Perbarui nilai m = m – 1
16. Akhir iterasi

Keluaran : bentuk cluster sampel objek sel darah merah yang
optimal
```

Gambar 3.12 *Pseudocode* dari *cluster validation* (Bagian 2)

Proses cluster validation dilakukan terus menerus hingga nilai dari m sama dengan dua, karena *cluster validation* hanya melakukan pengecekan pada sel darah merah bertumpuk. Variabel $\hat{m}$ adalah nilai dari *cluster validity index* yang telah dilakukan maksimasi seperti pada Persamaan (3.1).

$$\hat{m} = \arg \max_{m} \frac{\epsilon(m)}{\varphi(m)} \tag{3.1}$$

3.3.3.1 Fungsi H-Maxima

Pada proses *cluster validation*, baris nomor 4 hingga 7 terdapat perulangan mencari nilai *cluster* yang akan di proses menggunakan fungsi *H-Maxima*.

Fungsi *H_maxima* digunakan sebagai pembentukan batas *cluster* yang dijadikan sebagai inisialisasi *cluster* pada algoritma EM, penggunaan *H-maxima* menekan jumlah *regional maxima* yang tidak pasti atau semu dengan menggunakan parameter nilai *depth* yang terus bertambah. *Pseudocode* dari fungsi perulangan H-maxima ditunjukan pada Gambar 3.13 dan Gambar 3.14.

```
Masukan: variabel subImage, variabel nb

1.  inisialisasi nilai h-maxima cdhmax= -inf
2.  lakukan iterasi A hingga nilai h-maxima>=nilai cluster
```

Gambar 3.13 *Pseudocode* dari fungsi *H_Maxima* (Bagian 1)

```
3.  lakukan proses h-maxima transform dengan depth = depth +
    0.05,subImagehm=imhmax(subImage,depth)
4.  lakukan proses regional maxima dari hasil h-maxima
    transform, subzhm=imregionalmax(subImagehm)
5.  lakukan pencarian titik tengah cluster dan jumlah
    cluster yang terbentuk dari hasil proses regional
    maxima,
6.  mengubah nilai variabel cdhmax menjadi nilai jumlah
    cluster yang terbentuk
7.  kembali ke baris nomor 2
8.  jika nilai cdhmax >= nilai cluster maka lanjutkan ke
    proses ClusterValidity

Keluaran: titik tengah cluster yang baru
```

Gambar 3.14 *Pseudocode* dari fungsi *H_Maxima* (Bagian 1)

3.3.3.2 Fungsi Nearest Neighbour

Pada proses *cluster validation* baris nomor 10 terdapat fungsi konstruksi set R menggunakan *nearest neighbour*. Fungsi *nearest_neighbour* dilakukan untuk membentuk inisialisasi *cluster* dengan membandingkan jarak terdekat suatu sampel piksel terhadap titik piksel yang menjadi acuan, seperti yang telah dijelaskan pada subbab 2.1. *Pseudocode* dari fungsi *nearest neighbour* ditunjukan pada Gambar 3.15 dan Gambar 3.16.

```
Masukan: variabel subImage, variabel posa (anggota set),
variabel posb (titik cluster), variabel jumlah
cluster,variabel jumlah piksel

1.  Inisialisasi variabel seed yang mempunyai ukuran sama
    dengan subImage
2.  Lakukan iterasi A sebanyak jumlah piksel subImage
3.  Lakukan iterasi B sebanyak jumlah cluster
4.  Dilakukan inisialisasi awal variabel untuk menyimpan
    nilai minimal jarak minSeed = -inf.
5.  Dilakukan inisialisi nilai clusSeed = 0 yang akan
    dipakai untuk menyimpan nilai cluster suatu piksel.
```

Gambar 3.15 *Pseudocode* dari fungsi *nearest_neighbour* (Bagian 1)

```
6.  Dilakukan inisialisi nilai clusSeed = 0 yang akan
    dipakai untuk menyimpan nilai cluster suatu piksel.
7.  Dilakukan penghitungan jarak minimal posa – posb hingga
    iterasi B selesai
8.  Simpan nilai minimal dari penghitungan jarak, nilai
    cluster yang disimpan adalah indeks dari iterasi B saat
    mendapatkan jarak paling minimal
9.  Hitung semua jarak minimal anggota set hingga iterasi A
    berakhir
10. Simpan nilai cluster yang didapatkan pada matriks Seed,
    dan nilai cluster adalah nilai piksel Seed

Keluaran : matriks seed
```

Gambar 3.16 *Pseudocode* dari fungsi *nearest_neighbour* (Bagian 2)

Hasil dari fungsi *nearest_neighbour* kemudian menjadi masukan dari proses *clust assign* yang akan menghitung nilai parameter awal seperti rata-rata ($\boldsymbol{\mu}$) dan kovarians ($\boldsymbol{\Sigma}$) kemudian juga dilakukan penghitungan titik tengah *cluster* yang baru dan juga probabilitas prior **(**τ**).** yang akan menjadi masukan untuk algoritma EM.

3.3.3.3 Fungsi Clust Assign

Fungsi *clust_assign* merupakan proses lanjutan *dari nearest neighbour*. Pada proses ini dilakukan inisialisasi pencarian parameter yang tidak diketahui (*unknown parameter*) yaitu : rata-rata ($\boldsymbol{\mu}$), *prior probability* (τ), dan matriks kovarian ($\boldsymbol{\Sigma}$) yang dapat diperoleh dari sampel set yang telah terbentuk *cluster* nya seperti yang telah dijelaskan pada subbab 2.8. *Pseudocode* dari fungsi *clustAssgin* ditunjukan pada Gambar 3.17 dan Gambar 3.18.

```
Masukan: matriks seed, variabel jumlah cluster,variabel
jumlah piksel

1.  Inisialisasi variabel cluste yang akan dipakai untuk
    menyimpan titik tengah cluster
```

Gambar 3.17 *Pseudocode* dari fungsi *clust_assign* (Bagian 1)

```
2.  Inisialisasi variabel cellprob untuk menyimpan nilai
    probabilitas prior masing-masing cluster
3.  Lakukan iterasi A mulai dari i=1 hingga sebanyak jumlah
    cluster
4.  Lakukan pencarian nilai piksel apda matriks seed yang
    sama dengan nilai indeks iterasi cluster
5.  Hitung jumlah elemen dari cluster i kemudian dibagi
    dengan jumlah elemen set maka didapatkan probabilitas
    prior
6.  Hitung rata-rata posisi (x,y) dari cluster i sehingga
    didapatkan vektor rata-rata cluster i
7.  Hitung kovarians posisi (x,y) dari cluster I sehingga
    didapatkan matriks kovarians cluster i
8.  Iterasi A berakhir
9.  Simpan nilai seluruh nilai dalam satu struct bernama
    cell

Keluaran : struct cell yang berisi vektor rata-rata, matriks
kovarians dan vektor probabilitas prior
```

Gambar 3.18 *Pseudocode* dari fungsi *clust_assign* (Bagian 2)

3.3.3.4 Fungsi Likelihood

Pada fungsi *clust_assign* telah didapatkan parameter awal rata-rata ($\boldsymbol{\mu}$), kovarians ($\boldsymbol{\Sigma}$) dan probabilitas prior **(**τ**)**, maka perlu dilakukan proses penghitungan nilai *likelihood* terlebih dahulu dengan menggunakan fungsi *likelihood*. *Pseudocode* fungsi *likelihood* ditunjukan pada Gambar 3.19 dan Gambar 3.20.

```
Masukan: parameter probabilitas prior (p), rata-rata (M) dan
kovarians (V), sampel set (X), jumlah cluster (m)

1.  Inisialisasi nilai variabel total=0
2.  Lakukan iterasi A sebanyak jumlah piksel sampel set (X)
3.  Lakukan iterasi B sebanyak jumlah cluster (m)
4.  Hitung dl yaitu selisih dari X dan M
5.  Jika nilai determinan V pada indeks ke- j dari m != 0
```

Gambar 3.19 *Pseudocode* dari fungsi *likelihood* (Bagian 1)

```
6.  Maka nilai loglikelihood = log(p(j)) -
    0.5*log(det(V{j}))-0.5*(d1')*inv(V{j})*(d1)-log(2*pi);
7.  Jika nilai determinan V pada indeks ke- j dari m == 0
8.  Maka nilai loglikelihood = 0
9.  Nilai total=total+loglikelihood
10. Keluar dari iterasi B jika semua indeks telah terakses
11. Keluar dari iterasi A jika semua indeks telah terakses

Keluaran: nilai loglikelihood
```

Gambar 3.20 ***Pseudocode*** **dari fungsi** ***likelihood*** **(Bagian 2)**

Proses dilakukannya pencarian *likelihood* menggunakan parameter rata-rata (**$\boldsymbol{\mu}$**), *prior probability* (**т**), dan matriks kovarian (**$\boldsymbol{\Sigma}$**) dan digunakan pula penghitungan hasil perkalian *class conditional probability* dan *prior probability* dan dihitung sesuai jumlah sampel (**X**) seperti yang telah dijelaskan pada subbab 2. Pada baris ke 5 terdapat kondisi yang menyangkut tentang determinan dari matriks V, hal ini dilakukan agar proses tidak menghitung matriks singular yang dapat menyebabkan error karena tidak adanya nilai invers dari matriks V pada indeks ke – j tersebut.

3.3.3.5 Fungsi Expectation-Maximization

Nilai parameter awal yang didapat dari fungsi *clust_Assign* menjadi masukan bagi fungsi *expectation* dan *maximization* hingga didapatkan nilai parameter baru rata-rata (**$\boldsymbol{\mu}$**), kovarians (**$\boldsymbol{\Sigma}$**) dan probabilitas prior **(т)** yang memenuhi nilai *likelihood* yang optimal. *Pseudocode* algoritma EM ditunjukan pada Gambar 3.21. Untuk penjelasan dari proses *Expectation-Maximization* telah dibahas pada subbab 2.8.

```
Masukan: parameter probabilitas prior, rata-rata dan
kovarians, sampel set, jumlah cluster, nilai likelihood

1.  Inisialisasi nilai likelihood lama=inf
2.  Lakukan iterasi A hingga Likelihood baru > Likelihood
    lama
3.  Melakukan proses expectation dengan parameter
    probabilitas prior,rata-rata dan kovarians sehingga
    mendapatkan matriks dengan ukuran sampel set x banyak
    cluster
4.  Hasil dari proses expectation yaitu bayesian classifier
    menjadi masukan bagi proses maximization, kemudian
    didapatkan nilai parameter yang  baru
5.  Hitung likelihood yang baru dengan parameter nilai yang
    baru menggunakan proses likelihood
6.  Kembali ke proses no 1

Keluaran : nilai parameter yang optimal, likelihood yang
optimal, bayesian classifier yang optimal
```

Gambar 3.21 *Pseudocode* dari iterasi expectation_maximization

3.3.3.6 Fungsi Clust EM

Pada fungsi *clust_EM*, hasil *Bayesian classifier* dari fungsi EM yang optimal akan menjadi penentu suatu piksel masuk pada *cluster* mana dan ditunjukan pada proses *clust EM*. *Pseudocode* dari fungsi *clust_EM* ditunjukan pada Gambar 3.22 dan Gambar 3.23.

```
Masukan: bayesian classifier yang optimal, variabel sampel
set, matriks seed

1.  Inisialisasi matriks seed yang akan dijadikan
    penyimpanan nilai cluster piksel yang baru
2.  Lakukan iterasi A sebanyak jumlah elemen sampel set
3.  Lakukan inisialisasi variabel max = 0 dan maxclust = 0
    untuk menyimpan nilai maksimal bayesian classifier dan
    nilai indeks cluster
4.  Lakukan iterasi B sebanyak jumlah cluster
```

Gambar 3.22 *Pseudocode* dari fungsi *clust_EM* (Bagian 1)

```
5.  Melakukan operasi logika, membandingkan nilai maksimal
    bayesian classifier, kemudian jika ditemukan nilai yang
    paling maksimal, maka nilai maxclust = indeks dari
    iterasi B
6.  Keluar dari iterasi B,dan mengisi nilai seed dengan
    nilai cluster yang baru
7.  Keluar dari iterasi A
8.  Lakukan iterasi C sebanyak jumlah cluster
9.  Lakukan penghitungan probabilitas prior dan nilai titik
    tengah cluster
10. Keluar dari iterasi C

Keluaran: matriks seed yang berisi pembagian cluster, titik
tengah cluster dan probabilitas prior yang baru
```

Gambar 3.23 *Pseudocode* dari fungsi *clust_EM* (Bagian 2)

3.3.3.7 Fungsi Validate

Setelah didapatkan matriks *seed* yang berisi pembagian *cluster*, maka dilakukan tahapan *cluster validity index* yang membandingkan nilai $V(m)$ dan $V(max)$. Nilai dari $V(m)$ didapat dari proses *separation* dan *compactness* yang telah dijelaskan pada subbab 2.9. *Pseudocode* dari fungsi *validate* ditunjukan pada Gambar 3.24 dan Gambar 3.25.

```
Masukan: titik tengah cluster (centclust), titik tengah
sampel (centro), bayesian classifier (pst), sampel set (X)

1.  %separation
2.  Inisialisasi nilai totsep=0
3.  Lakukan iterasi A sebanyak jumlah cluster
4.  Menghitung nilai bet yaitu nilai centclust – centro
5.  Hitung nilai perkalian vector bet, sehingga menjadi
    matriks bet2
6.  Inisialisasi nilai tsep=0
7.  Lakukan iterasi B sebanyak jumlah piksel sampel set
8.  Kalikan nilai pst dengan bet2 dan menghasilkan
    tsep=tsep+pst*bet2
9.  Keluar dari iterasi B jika indeks telah memenuhi jumlah
    piksel
```

Gambar 3.24 *Pseudocode* dari fungsi *validate* (Bagian 1)

```
10. Nilai totsep=totsep+tsep
11. Keluar dari iterasi A
12. Lakukan trace pada totsep dan simpan pada variabel seps.
13. %compactness
14.
15. Inisialisasi nilai totcomp=0
16. Lakukan iterasi A sebanyak jumlah cluster
17. Menghitung nilai wit yaitu nilai X - centro
18. Hitung nilai perkalian vector wit, sehingga menjadi
    matriks wit2
19. Inisialisasi nilai tcomp=0
20. Lakukan iterasi B sebanyak jumlah piksel sampel set
21. Kalikan nilai pst dengan wit2 dan menghasilkan
    tcomp=tcomp+pst*wit2
22. Keluar dari iterasi B jika indeks telah memenuhi jumlah
    piksel
23. Hitung nilai comp2 yaitu tcomp dibagi dengan jumlah pst
24. Hitung nilai totcomp yaitu totcomp+trace(comp2)
25. Keluar dari iterasi A jika indeks telah memenuhi jumlah
    piksel
26. Simpan hasil pada variabel comps
27. Hitung nilai V(m)=seps/comps
28. Jika nilai V(m)>Vmax
29. Vmax=Vm
30. Mclust=nilai cluster sekarang
31. Nilai seed yang baru disimpan pada seedOp

Keluaran : seedOp yaitu yang baru, nilai V(m) yang baru
```

Gambar 3.25 ***Pseudocode*** **dari fungsi** ***validate*** **(Bagian 2)**

3.3.3.8 Fungsi Colouring

Setelah didapatkan bentuk *cluster* untuk masing-masing objek sel darah merah, maka dilakukan proses pewarnaan anggota *cluster* untuk masing-masing objek sel darah merah, sehingga akan mempermudah visualisasi pemisahan sel darah merah yang bertumpuk secara kasat mata, *Pseudocode* pewarnaan visualisasi ditunjukan pada Gambar 3.26.

```
Masukan: matriks seed sel darah merah yang berisi pembagian
cluster

1.  Inisialisasi citra RGB kosong dengan nilai = 0
2.  Lakukan iterasi A sebanyak jumlah objek sel darah merah
3.  Melakukan proses perubahan dari citra biner labelling
    menjadi RGB dengan fungsi MATLAB label2rgb() dan menjadi
    citra objek sel darah merah yang baru dengan warna
    berbeda untuk tiap cluster yang terbentuk,
    mask=label2rgb(seed)
4.  Kemudian lakukan penggabungan antara citra RGB kosong
    dengan citra objek sel darah merah yang telah diwarnai
    dengan citra RGB awal.
5.  Keluar dari iterasi A.

Keluaran: Citra gabungan sel darah merah yang bertumpuk dan
yang tidak bertumpuk yang dibedakan dengan warna
```

Gambar 3.26 *Pseudocode* dari fungsi *colouring*

Hasil *output* dari fungsi *colouring* akan berupa citra sel darah yang baru dengan pewarnaan pada sel darah merah bertumpuk agar terlihat pemisahan dari sel darah merah bertumpuk.Hasil *output* pada fungsi *colouring* merupakan akhir dari proses identifikasi sel bertumpuk.

BAB IV
IMPLEMENTASI

Pada bab ini akan dijelaskan tahap implementasi Identifikasi sel darah merah bertumpuk menggunakan *unsupervised Bayesian classification* pada citra mikroskopik sel darah merah yang meliputi lingkungan implementasi dan potongan kode program dengan menggunakan perangkat MATLAB.

4.1 Lingkungan Implementasi

Pada penelitian ini, tahapan implementasi dan pengujian dilakukan dengan menggunakan perangkat keras dan perangkat lunak yang digunakan untuk proses identifikasi sel darah merah bertumpuk menggunakan *unsupervised Bayesian classification* pada citra mikroskopis sel darah merah.Perangkat keras yang digunakan mempunyai spesifikasi sebagai berikut:

- Laptop ASUS A450L
  - Windows 8.1 64-bit.
  - Prosesor Intel(R) Core(TM) i5-4200U CPU @ 1.60GHz.
  - RAM : 4 GB.

  Sedangkan perangkat lunak yang digunakan adalah:
- Windows 8.1 64-bit sebagai sistem operasi.
- MATLAB 8.1 R2013a sebagai aplikasi pendukung.

4.2 Implementasi

Pada bagian ini akan dijelaskan mengenai implementasi kode program yang terdapat dalam perangkat lunak mengenai proses identifikasi sel darah merah bertumpuk menggunakan *unsupervised Bayesian classification* pada citra mikroskopis sel darah merah. dan implementasi setiap proses yang telah dijelaskan pada bab-bab sebelumnya.

4.2.1 Implementasi Tahapan Preprocessing

Pada Tahapan *preprocessing*, dilakukan lima proses yaitu *Green Component*, *Median Filtering*, *Gray Thresholding*, *Edge detection (canny)*, dan *Merging* seperti yang telah dijelaskan pada subbab 3.3.1.

```
1. function inputims(IN);
2.  ik=imread(IN);
3.  J=ik(:,:,2);
4.  J2=medfilt2(J);
5.  r1=im2bw(J2,graythresh(J2));
6.  nr1=~r1;
7.  r=bwareaopen(nr1,100);
8.  e=edge(J2,'canny');
9.  BW2=bwareaopen(e,5);
10. s=BW2+r;
11. sn=imfill(s,'holes');
12. fil=bwareaopen(sn,100);
```

Gambar 4.1 Kode sumber tahapan *Preprocessing*

Pada Gambar 4.1, implementasi tahapan *Preprocessing* menggunakan pustaka dari MATLAB. Implementasi *Green Component* dapat dilihat pada baris ke-3 yaitu mengambil nilai *green channel*, untuk *median filtering* dilakukan dengan fungsi *medfilt2*. Implementasi *thresholding* dan merubah ke biner dilakukan dengan menggunakan fungsi *im2bw* dan *graythresh*. *Edge detection canny* diimplementasikan menggunakan fungsi *edge*(image*,'canny'*), pengisian lubang pada citra menggunakan *imfill*, kemudian *Merging* dilakukan dengan cara menambahkan dua citra yang memiliki ukuran matriks yang sama menjadi satu citra.

4.2.2 Implementasi Tahapan Segmentasi Sel Darah Merah

Pada tahapan segmentasi sel darah merah, dilakukan dua proses yaitu proses penghilangan sel darah putih dan platelet dan proses penutupan lubang yang berada di pinggir citra. Hasil dari

dua proses tersebut nantinya akan membentuk citra sel darah merah yang lebih optimal.

Proses penghilangan sel darah putih dan platelet dilakukan untuk membuat citra sel darah merah yang akan diproses seperti yang telah dijelaskan pada subbab 3.3.2 mengenai proses penghilangan citra sel darah putih dan platelet. Kode sumber untuk penghilangan sel darah putih dapat dilihat pada Gambar 4.2.

```
1. rPlane=ik(:,:,3)-0.5*(ik(:,:,1))-0.5*(ik(:,:,2));
2.  BW=rPlane > 29;
3.  BW2=bwareaopen(BW,100);
4.  stats=regionprops(BW2, 'EquivDiameter');
5.  allDiam=[stats.EquivDiameter];
6.  rata=round(mean(allDiam)/17);
7.    if isequalwithequalnans(rata,NaN)==1
8.       rata=1;
9.    end
10.  se=strel('disk',rata);
11.  BW2=imdilate(BW2,se);
12.  burnedImage=fil-BW2;
13.  fil=imfill(burnedImage,'holes');
14.  fil=bwareaopen(fil,100);
```

Gambar 4.2 Kode sumber proses penghilangan sel darah putih dan platelet

Kemudian terdapat proses untuk penutupan lubang pada tepi citra dengan menggunakan pengecekan tepi dan tetangga piksel sehingga mengubah nilai piksel menjadi satu seperti yang telah dijelaskan pada subbab 3.3.1, dengan cara melakukan pengecekan terhadap tepi dari piksel tersebut, apakah piksel mendekati tepi tersebut mempunyai tetangga piksel yang bukan *background*. Kode sumber untuk proses penutupan lubang dapat dilihat pada Gambar 4.3 dan Gambar 4.4.

```
1. s=regionprops(fil,'all');
2. fil2=fil;
3. xx=1;
4. for k = 1 : numel(s)
5.     param=[s.Perimeter];
6.     minParam=round(min(param));
7.     maxParam=round(max(param));
8.     rtParam=round((minParam+maxParam)/2);
9.      siz=size(fil);
10.      cekY=length(find(s(k).PikselList(:,1)==1))>0;
11.      cekX=length(find(s(k).PikselList(:,2)==1))>0;
12.    cekY2=length(find(s(k).PikselList(:,1)==siz(:,2)))>0;
13.    cekX2=length(find(s(k).PikselList(:,2)==siz(:,1)))>0;
14.       if(((cekX )|(cekY )|(cekX2
    )|(cekY2))&((s(k).Perimeter>minParam))&(s(k).Perimeter<rt
    Param))
15.          if(cekY)
16.              cek(xx)=k;xx=xx+1;
17.              jb=find(s(k).PikselList(:,1)==1);
18.              ljb=length(jb);
19.              fmin=s(k).PikselList(jb(1),2);
20.              fmax=s(k).PikselList(jb(ljb),2);
21.              for jj=fmin:fmax
22.                 fil2(jj,1)=1;
23.              end
24.          end
25.          if(cekX)
26.               cek(xx)=k;xx=xx+1;
27.               jb=find(s(k).PikselList(:,2)==1);
28.               ljb=length(jb);
29.               fmin=s(k).PikselList(jb(1),1);
30.               fmax=s(k).PikselList(jb(ljb),1);
31.               for jj=fmin:fmax
32.                   fil2(1,jj)=1;
33.               end
34.          end
35.          if(cekY2)
36.               cek(xx)=k;xx=xx+1;
37.               jb=find(s(k).PikselList(:,1)==siz(:,2));
38.               ljb=length(jb);
39.               fmin=s(k).PikselList(jb(1),2);
40.               fmax=s(k).PikselList(jb(ljb),2
```

Gambar 4.3 Kode sumber proses penutupan lubang pada pinggir citra biner sel darah merah (Bagian 1)

```
41.                 for jj=fmin:fmax
42.                     fil2(jj,siz(:,2))=1;
43.                 end
44.             end
45.             if(cekX2)
46.                 cek(xx)=k;xx=xx+1;
47.                 jb=find(s(k).PikselList(:,2)==siz(:,1));
48.                 ljb=length(jb);
49.                 fmin=s(k).PikselList(jb(1),1
50.                 fmax=s(k).PikselList(jb(ljb),1
51.                 for jj=fmin:fmax
52.                     fil2(siz(:,1),jj)=1;
53.                 end
54.             end
55.             fil2=imfill(fil2,'holes');
56.         end
57. end
```

Gambar 4.4 Kode sumber proses penutupan lubang pada pinggir citra biner sel darah merah (Bagian 2)

4.2.3 Implementasi Tahapan Identifikasi Sel Bertumpuk

Beberapa Proses dilakukan pada tahapan identifikasi sel bertumpuk, mulai dari proses *distance transform* hingga proses *cluster validation*.

```
1. s=fil2;
2.  jumT=numel(regionprops(bwlabel(s),'all'));
3.  N=bwlabel(s);
4.  W=bwdist(~N);
5.  W2=bwlabel(W);
6.  ovlpdist=regionprops(W2,'all');
7.  for n=1:numel(ovlpdist)
8.      centinit{n}=[];
9.      a(:,n) = [ovlpdist(n).Centroid];
10.     c = a(1,n);
11.     r = a(2,n);
12.
```

Gambar 4.5 Kode sumber proses pembuatan masing-masing objek sel darah merah (Bagian 1)

```
13.     BWs{n}=bwselect(N,c,r);
14.     subIm=~BWs{n};
15.     subImage{n}=bwdist(~subIm);
16.
17.     cropimage(:,:,1)=BWs{n}*255;
18.     cropimage(:,:,2)=BWs{n}*255;
19.     cropimage(:,:,3)=BWs{n}*255;
20.
21.     sImage{n}=uint8(cat(3, cropimage(:,:,1),
    cropimage(:,:,2), cropimage(:,:,3)));
22.
23.
24.     subz{n}=imregionalmax(subImage{n});
25.
26.
27.     subImage1{n}=bwdist(~subIm);
28.     subz1{n}=imregionalmax(round(subImage1{n}));
29.     subz1lb{n}=bwlabel(subz1{n});
30.     ovlpdistg=regionprops(subz1lb{n},'all');
31.     ovlpd{n}=ovlpdistg;
32.     mn=[];
33.     for o=1:numel(ovlpd{n})
34.         nm=ovlpdistg(o).Centroid;
35.         mn(o,1)=mean(nm(:,1));
36.         mn(o,2)=mean(nm(:,2));
37.     end
38.     mnm=[mean(mn(:,1)) mean(mn(:,2))];
39.     centinit{n}=mn;
40.     centroids2{n} = mnm';
41. end
```

Gambar 4.6 Kode sumber proses pembuatan masing-masing objek sel darah merah (Bagian 2)

Pada Gambar 4.5 dan Gambar 4.6 ditampilkan kode sumber dilakukannya proses *distance transform* menggunakan fungsi *bwdist*, *regional maxima* menggunakan fungsi *imregionalmax*, pelabelan menggunakan fungsi *bwlabel*, permotongan objek menjadi sampel citra baru menggunakan *bwselect* dan penentuan titik tengah suatu sampel objek menggunakan *regionprops centroid* yang akan digunakan untuk penghitungan *separation* dan *compactness* pada proses *cluster validation*.

```
1. for i=1:jumT
2.       [y x]=find(subImage{i}>0);
3.        pospixl{i}=[x y];
4.        pospixl{i}=transpose(pospixl{i});
5.        pospixlt{i}=transpose(pospixl{i});
6.        posa = [x y];
7.        na=numel(posa)/2;
8.        [y1 x1]=find(subz{i}>0);
9.        posb = centinit{i};
10.        nb=numel(posb)/2;
11.        mclustest=nb;
12.        mclust{i}=nb;
13.        mclustawal{i}=nb;
14.        Vmax{i}=-Inf;
15.        oq{i}=0;
16.        turn=1;
17.        seedOp{i}=[];
18.        mask{i}=[];
19.        depth=0;
20.        cdhmax{i}=-inf;
21.        mh=centinit{i};
22.
23.          cluster_validation()
24.          colouring()
25. end
```

Gambar 4.7 Kode sumber proses setiap objek sel darah merah

Pada Gambar 4.7 dilakukan iterasi untuk memproses semua objek sel darah merah yang ditemukan setelah diproses pada pelabelan sebelumnya, kemudian setiap sampel objek sel tersebut dilakukan proses *cluster validation* dengan fungsi *cluster_validation*, proses yang dilakukan setelah *cluster validation* adalah membuat citra pembentukan sel darah merah bertumpuk dengan memberi warna untuk setiap *cluster* yang terbentuk dengan menggunakan fungsi *label2rgb*.

Pembahasan selanjutnya akan membahas lebih detil tentang proses yang dilakukan pada fungsi *cluster_validation* yang sesuai dengan perancangan fungsi yang dijelaskan pada subbab 3.3.3.

4.2.3.1 Implementasi Fungsi H-Maxima

Proses dengan *H-maxima* digunakan sebagai penentuan batas *cluster* yang digunakan sebagai inisialisasi pada algoritma EM, penggunaan *H-maxima* menekan jumlah *regional maxima* yang tidak pasti atau semu. Sehingga dapat mempercepat proses karena dilakukan pengabaian jumlah *cluster* yang akan dihitung. Pada Gambar 4.8 ditampilkan kode sumber proses *H-maxima*.

```
1. while cdhmax{i} < mclustest
2.       depth=depth+0.05;
3.       subImagehm=imhmax(round(subImage1{i}),depth)
4.       subzhm=imregionalmax(subImagehm);
5.       subzhmlbl=bwlabel(subzhm);
6.       ovlpdisthm=regionprops(subzhmlbl,'all');
7.       mh=[];
8.       for o=1:numel(ovlpdisthm)
9.           hm=ovlpdisthm(o).Centroid;
10.          mh(o,1)=mean(hm(:,1));
11.          mh(o,2)=mean(hm(:,2));
12.      end
13.      cdhmax{i}=numel(ovlpdisthm);
14. end
15. centinithm{i}=mh;
16. posb=centinithm{i};
17. if cdhmax{i} == mclustest
18.    %proses selanjutnya
```

Gambar 4.8 Kode sumber fungsi H_maxima

Dilakukan perulangan dengan kondisi pada baris pertama untuk menekan jumlah inisialisasi *cluster* yang akan dipakai pada proses pembuatan *cluster*. Implementasi H-maxima menggunakan fungsi *imhmax*, kemudian dilakukan penyimpanan citra biner yang baru dan dilakukan proses pencarian anggota kelompok *cluster* seperti yang dijelaskan pada subbab 3.3.3.

4.2.3.2 Implementasi Fungsi Nearest Neighbour

Proses *nearest neighbour* dilakukan untuk membentuk inisialisasi *cluster* dengan membandingkan jarak terdekat suatu sampel piksel terhadap titik piksel yang menjadi acuan, seperti yang telah dijelaskan pada subbab 2.7. Pada Gambar 4.9 ditampilkan kode sumber fungsi *nearest_neighbour*.

```
1. function [A seed] =
    nearest_neighbour(na,nb,posa,posb,seed,subImage)
2. for j=1:na
3.      minseed=Inf;
4.      clusseed=0;
5.      ba=0;
6.      for l=1:nb
7.          p = power((posa(j,1)-posb(l,1)),2);
8.          q = power((posa(j,2)-posb(l,2)),2);
9.          ba = sqrt(p + q);
10.             if minseed>ba
11.             minseed=ba;
12.             clusseed=l;
13.            end
14.          seed(posa(j,2),posa(j,1)) = clusseed;
15.     end
16.     xi=posa(j,1);
17.     yi=posa(j,2);
18.     pix2=subImage(yi,xi);
19.     A(j)=pix2;
20. end
```

Gambar 4.9 Kode sumber fungsi *nearest_neighbour*

Implementasi dari fungsi *nearest_neighbour* berdasarkan dari fungsi yang telah dijelaskan pada subbab 3.3.3.2, dilakukan iterasi sebanyak jumlah piksel objek yang sedang diproses, kemudian dilakukan iterasi sebanyak jumlah *cluster* yang akan diproses.

Dilakukan proses penghitungan jarak terdekat menggunakan *euclidean distance* yang diimplementasikan pada baris ke – 7 hingga baris ke – 9, jika nilai jarak tersebut mencapai nilai minimal, maka nilai dari variabel *clusseed* adalah nilai indeks

ke – l saat ditemukannya jarak terdekat, kemudian hasil dari *clusseed* disimpan pada matriks *seed* yang baru.

4.2.3.3 Implementasi Fungsi Clust Assign

Proses *clust assign* merupakan proses dilakukannya inisialisasi pencarian parameter yang tidak diketahui (unknown parameter) yaitu : rata-rata ($\boldsymbol{\mu}$), *prior probability* (τ), dan matriks kovarian ($\boldsymbol{\Sigma}$) yang dapat diperoleh dari sampel set yang telah terbentuk *cluster* nya seperti yang telah dijelaskan pada subbab 2. Pada Gambar 4.10 dan 4.11 ditampilkan kode sumber fungsi *clust_assign*.

```
1. function [cell] =
    clust_assign(na,nb,posa,posb,seed,subImage,A)
2.  clust= [];
3.  cluste= [];
4.  centclus= [];
5.
6.  for l=1:nb
7.      post=[];
8.      [y x]=find(seed==l);
9.      pi=[x y];
10.      p=numel(pi)/2;
11.      pi=transpose(pi);
12.      prob=p/na;
13.
14.      cell.cells(l)=l;
15.      cell.probs(l)=prob;
16.      Bs{l}=cov(pi(1,:),pi(2,:));
17.
18.      clust{l}=struct('cell',l,'pos',[x y],'probs',prob);
19.      cluste{l}=struct('pos',[x y]);
20.
21.      po=[mean(x) mean(y)];
22.      centcluspos(1,l) = po(1);
23.      centcluspos(2,l) = po(2);
24.  end
```

Gambar 4.10 Kode sumber fungsi *clust_assign* (Bagian 1)

```
25. cell.centpos=centcluspos;
26.  cell.pos=cluste;
27.  cell.cluster=seed;
28.  cell.cova=Bs;
29.  cell.probs=transpose(cell.probs);
30.
31.  end
```

Gambar 4.11 Kode sumber fungsi *clust_assign* (Bagian 2)

Dilakukan proses pencarian parameter rata-rata ($\boldsymbol{\mu}$), *prior probability* (τ), dan matriks kovarian ($\boldsymbol{\Sigma}$) untuk setiap jumlah *cluster* yang ada karena masing – masing parameter berkesinambungan dengan jumlah *cluster* yang merupakan variabel laten seperti yang dijelaskan pada subbab 2.8. Kemudian hasil parameter tersebut masing – masing disimpan pada variabel cell.centpos, cell.probs, dan cell.cova.

4.2.3.4 Implementasi Fungsi Likelihood

Proses *likelihood* merupakan proses dilakukannya pencarian *likelihood* dengan parameter rata-rata ($\boldsymbol{\mu}$), *prior probability* (τ), dan matriks kovarian ($\boldsymbol{\Sigma}$) dan digunakan penghitungan hasil perkalian *class conditional probability* dan *prior probability* dan dihitung sesuai jumlah sampel ($\mathbf{X}$) seperti yang telah dijelaskan pada subbab 2.8 Pada Gambar 4.12 ditampilkan kode sumber fungsi *likelihood.*

```
1. function L = likelihood(X,k,W,M,V)
2.  [n,e] = size(X);
3.  tot=0;
4.  for i=1:e
5.      for j=1:k
6.          d1=X(:,i)-M(:,j);
```

Gambar 4.12 Kode sumber fungsi *likelihood* (Bagian 1)

```
7.          if det(V{j}) ~= 0
8.             pn=log(W(j)) - 0.5*log(det(V{j}))-
9.             0.5*(d1')*inv(V{j})*(d1)-log(2*pi);
10.         else
11.            pn=0;
12.         end
13.         tot=tot+pn;
14.      end
15.  end
16.  L=tot;
```

Gambar 4.13 Kode sumber fungsi *likelihood* (Bagian 2)

Pada iterasi kedua terdapat fungsi pengambilan keputusan jika determinan tidak sama dengan nol maka dilakukan penghitungan, hal ini dilakukan untuk mencegah terbentuknya matriks singular, karena matriks singular tidak memiliki nilai invers, dan hasil penghitungan nanti nilai dari *likelihood* akan menjadi tidak terdefinisi.

4.2.3.5 Implementasi Fungsi Expectation-Maximization

Proses *expectation-maximization* merupakan proses dilakukannya pencarian *likelihood* dengan parameter rata-rata ($\boldsymbol{\mu}^{(t)}$), *prior probability* ($\tau^{(t)}$), dan matriks kovarian ($\boldsymbol{\Sigma}^{(t)}$) kemudian dilakukan penghitungan ulang dan didapatkan parameter rata-rata ($\boldsymbol{\mu}^{(t+1)}$), *prior probability* ($\tau^{(t+1)}$), dan matriks kovarian ($\boldsymbol{\Sigma}^{(t+1)}$) yang baru, sehingga nilai *likelihood* berubah dan dapat dibandingkan dengan *likelihood* sebelumnya.

Proses *expectation-maximization* memakan banyak waktu dan membutuhkan banyak iterasi, dikarenakan pencarian nilai *likelihood* yang optimal, maka dari itu ditambahkan batas banyak iterasi yang dilakukan, agar mempercepat proses. Pada Gambar 4.14 ditampilkan kode sumber fungsi *expectation_maximization.*

```
1. Ln=likelihood(pospixl{i},mclustest,po,mu,v);
2.
3. Lo=Ln-1;
4. niter=0;
5. maxiter=1000;
6. pon=po;
7. mn=mu;
8. vn=v;
9.
10. while (Lo<Ln) & (niter<=maxiter)
11.
12.      E=expectation(mn,vn,pon,pospixl{i},mclustest);
13.      [pon,mn,vn]=maximization(pospixl{i},mclustest,E);
14.
15.      Lo=Ln;
16.
17.      Ln=likelihood(pospixl{i},mclustest,pon,mn,vn);
18.
19.      niter=niter+1;
20.      op{i}=niter;
21. end
22.
```

Gambar 4.14 Kode sumber fungsi *expectation_maximization*

Pada proses *expectation* parameter yang dihitung adalah rata-rata (***μ***), *prior probability* (τ), matriks kovarian (***Σ***), sampel set (**X**), dan jumlah *cluster* (*m)*. Hasil dari proses expectation berupa sebuah matriks probabilitas seperti yang telah dijelaskan pada subbab 2.8 Pada Gambar 4.15 ditampilkan kode sumber proses *expectation.*

```
1. function E = expectation(m,v,g,x,nb)
2.
3.  E = zeros(numel(x)/2,nb);
4.  for i=1:numel(x)/2
5.     tot=0;
6.        for j=1:nb
7.            h=0;
8.            amp=0;
9.            d = x(:,i)-m(:,j)
10.            if det(v{j}) ~=
11.               h=d'*inv(v{j})*d;
12.               amp=g(j)/sqrt(2*pi*det(v{j}));
13.               E(i,j)=amp*exp(-0.5*h);
14.            else
15.            E(i,j)=0;
16.            end
17.          tot=tot+E(i,j);
18.        end
19.   E(i,:) = E(i,:)/tot;
20. end
```

Gambar 4.15 Kode sumber fungsi expectation

Hasil dari proses *expectation* adalah matriks E yang berukuran jumlah piksel pada objek (*n*) dan jumlah *cluster* (*m*).matriks E merupakan *posterior probability* yang merupakan *Bayesian classifier* untuk penentuan *cluster*.

Pada proses *maximization* parameter yang dihitung adalah hasil dari expectation yang berupa matriks (*E*), sampel set (**X**), dan jumlah *cluster* (*m)*. Hasil dari proses *maximization* berupa parameter rata-rata ($\boldsymbol{\mu}^{(t+1)}$), *prior probability* ($\tau^{(t+1)}$), dan matriks kovarian ($\boldsymbol{\Sigma}^{(t+1)}$) yang baru, seperti yang telah dijelaskan pada subbab 2. Pada Gambar 4.16 ditampilkan kode sumber fungsi *maximization.*

Hasil dari proses *maximization* adalah parameter rata-rata ($\boldsymbol{\mu}^{(t+1)}$), *prior probability* ($\tau^{(t+1)}$), dan matriks kovarian ($\boldsymbol{\Sigma}^{(t+1)}$) yang baru, yang kemudian akan dihitung kembali pada proses expectation jika hasil *likelihood* yang baru tidak memenuhi kondisi seperti pada Gambar 4.10.

```
1. function [W,M,V] = maximization(X,k,E)
2.  [n,d]=size(E);
3.  p1=sum(E(:,1));
4.  p2=sum(E(1,:));
5.
6.  allp=sum(sum(E))
7.  Mu=zeros(2,d);
8.  for j=1:d
9.     p(j)=sum(E(:,j))/allp;
10.     for i=1:n
11.         kl=E(i,j)*X(:,i);
12.         Mu(1,j)=Mu(1,j)+kl(1);
13.         Mu(2,j)=Mu(2,j)+kl(2);
14.     end
15.     Mu(1,j)=Mu(1,j)/sum(E(:,j));
16.     Mu(2,j)=Mu(2,j)/sum(E(:,j));
17.     Va=zeros(2,2);
18.     for i=1:n
19.         d=X(:,i)-Mu(:,j);
20.         Va=Va+E(i,j)*d*d';
21.     end
22.     Va=Va/sum(E(:,j));
23.     Vu{j}=Va;
24.   end
25.  W=p';
26.  M=Mu;
27.  V=Vu;
```

Gambar 4.16 Kode sumber fungsi *maximization*

4.2.3.6 Implementasi Fungsi Clust EM

Pada proses c*lust EM*, dilakukan proses *clustering* menggunakan *bayes classifier*, yang merupakan probabilitas posterior ($T_{j,i}$) pada proses *expectation* seperti yang telah dijelaskan pada subbab 2.8. Suatu piksel ($\boldsymbol{X}_i$) dikelompokan pada *cluster j* jika piksel tersebut memiliki nilai probabilitas posterior ($T_{j,i}$) yang paling maksimal dibandingkan *cluster* lainnya. Output dari c*lust EM* adalah sel darah merah yang telah di kelompokan dan center dari setiap *cluster* yang ditemukan. Pada Gambar 4.17 ditampilkan kode sumber fungsi c*lust_EM*.

```
1. function [seed,prob,posc] = clustEM(posa,E,seed)
2. [n,d]=size(E);
3. for i=1:n
4.     max=0;
5.     maxclust=0;
6.     for j=1:d
7.         if E(i,j)>max
8.             max=E(i,j);
9.             maxclust=j;
10.         end
11.     end
12.     seed(posa(i,2),posa(i,1)) = maxclust;
13. end
14.
15. for l=1:d
16.     [y x]=find(seed==l);
17.     pi=[x y];
18.     p=numel(pi)/2;
19.     prob(l)=p/n;
20.     posc(l,1)=mean(x);
21.     posc(l,2)=mean(y);
22. end
```

Gambar 4.17 Kode sumber fungsi *clust_EM*

Hasil dari proses *clust EM* adalah matriks *seed* yang telah dikelompokan berdasarkan *cluster* menggunakan probabilitas posterior, kemudian dilakukan penghitungan probabilitas prior dan titik tengah masing-masing *cluster*.

4.2.3.7 Implementasi Fungsi Validate

Pada proses *validate*, dilakukan proses mencari nilai *cluster validity index*, yang didapat dari hasil *separation* dibagi dengan *compactness* seperti yang dijelaskan pada subbab 2.9. Nilai *cluster validity index* nanti akan menjadi penentu apakah nilai suatu *cluster* lebih baik dibandingkan *cluster* lainnya. Hasil dari *cluster* validity adalah nilai *maximum cluster* dan *cluster* yang terbentuk Pada Gambar 4.18 ditampilkan kode sumber fungsi v*alidate*.

```
%%Validate


%Separation
totsep=0;
            for j=1:mclustest
                bet=cell{i}.centclust(j)-cell{i}.centro;
                bet2=bet*bet';
                tsep=0;
                 for k=1:na
                     tsep=tsep+cell{i}.pst(k,j)*bet2;
                 End
                 totsep=totsep+tsep;
             End
             seps{i}=trace(totsep);

%Compactness
             totcomp=0;
             for j=1:mclustest
                 wit=cell{i}.postpixl(:,j)-cell{i}.centro;
                 wit2=wit*wit';
                 tcomp=0;
                 for k=1:na
                     tcomp=tcomp+cell{i}.pst(k,j)*wit2;
                 end;
                 comp2=tcomp/sum(cell{i}.pst(:,j));
                 totcomp=totcomp+trace(comp2);
             end;
             comps{i}=totcomp;

Vm{i}(turn)=seps{i}/comps{i};
             if(Vm{i}(turn)>=Vmax{i})
                 oq{i}=oq{i}+1;
                 Vmax{i}=Vm{i}(turn);
                 mclust{i}=mclustest;
                 seedop{i}=seed{i};
                 cell{i}.probsOp=cell{i}.probclust;
                 cell{i}.centclustOp=cell{i}.centclust;
             end
```

Gambar 4.18 Kode sumber proses *validate*

Nilai dari *cluster validity index* akan menentukan jumlah *cluster* dan bentuk *cluster* yang optimal seperti yang dijelaskan pada subbab 2.9, dan hasil dari bentuk *cluster* yang optimal disimpan pada matriks *seedop* yang kemudian akan diproses

menjadi citra visualisasi pemisahan sel darah merah yang bertumpuk pada matriks *seedop* tersebut dan dilakukan penggabungan citra.

4.2.3.8 Implementasi Fungsi Colouring

Pada proses *colouring*, dilakukan proses pewarnaan anggota *cluster* untuk masing-masing objek sel darah merah, sehingga menghasilkan citra visualisasi pemisahan sel darah merah yang bertumpuk dan didapakan citra gabungan dengan cara menggabungkan citra sel darah awal dengan hasil visualisasi pemisahan sel darah merah yang bertumpuk. Pada Gambar 4.19 ditampilkan kode sumber proses c*olouring*.

```
1.          if mclust{i}>1
2.             mask{i}=label2rgb(seedOp{i},'jet','k')
3.
4.             clustimage = clustimage + mask{i};
5.             clustimage2 = clustimage2 + mask{i};
6.
7.             else
8.             mask{i}=sImage{i};
9.             clustimage = clustimage + mask{i};
10.          end
```

Gambar 4.19 Kode sumber fungsi colouring

Matriks *sImage* adalah citra RGB hasil konversi citra biner sel darah merah yang telah didapatkan pada tahapan segmentasi citra pada Gambar 4.6

Terdapat dua hasil dari proses *colouring*, yaitu citra visualisasi pemisahan sel darah merah bertumpuk yang bernama *clustimage* dan citra gabungan yang bernama *clustimage2*, untuk citra visualisasi pemisahan sel darah merah bertumpuk, sel darah merah yang dianggap tidak bertumpuk memiliki warna putih dan sel darah merah yang dianggap bertumpuk memiliki berbagai ragam warna bergantung dari jumlah *cluster* yang dideteksi.

BAB V
HASIL UJI COBA DAN EVALUASI

Pada bab ini akan dijelaskan uji coba dan evaluasi yang dilakukan pada identifikasi sel darah merah bertumpuk menggunakan *unsupervised Bayesian classification*. Pembahasan yang dilakukan meliputi lingkungan pengujian, data uji coba, skenario uji coba, dan hasil uji coba.

5.1 Lingkungan Pengujian

Proses uji coba dan evaluasi Tugas Akhir ini digunakan perangkat keras berupa laptop dengan spesifikasi sebagai berikut:

- Laptop ASUS A450L
  - Windows 8.1 64-bit.
  - Prosesor Intel(R) Core(TM) i5-4200U CPU @ 1.60GHz.
  - RAM : 4 GB.

  Sedangkan perangkat lunak yang digunakan adalah:
- Windows 8.1 64-bit sebagai sistem operasi.
- MATLAB 8.1 R2013a sebagai aplikasi pendukung.

5.2 Data Uji Coba

Data masukan yang digunakan dalam uji coba ini adalah dataset penelitian pembimbing Tugas Akhir, seperti yang sudah dijelaskan pada subbab 3.2. Dataset tersebut terdiri dari 10 citra mikroskopik sel darah.

Data keluaran hasil uji coba adalah citra visualisasi pemisahan sel darah merah yang bertumpuk untuk tiap sel darah merah.

5.3 Skenario Uji Coba

Pada subbab ini akan dijelaskan mengenai skenario uji coba yang dilakukan untuk mengetahui kinerja identifikasi sel darah merah bertumpuk menggunakan *unsupervised Bayesian*

classification. Skenario uji coba akan dibagi menjadi 2 skenario dengan rincian sebagai berikut:

1. Skenario Uji coba perbandingan dengan segmentasi citra manual, segmentasi pada tahap ini adalah melakukan pemisahan sel darah merah dengan *background* yang tidak diperlukan seperti sel darah merah dan platelet. Citra hasil segmentasi manual (*ground truth*) akan dianggap sebagai hasil segmentasi yang benar. Sehingga akurasi segmentasi dari metode yang diusulkan dapat dibandingkan.
2. Skenario uji coba akurasi segmentasi sel darah merah bertumpuk melalui penghitungan jumlah *cluster* dengan pengecekan *oversegmentation*, *undersegmentation* serta *correctly segmented* berdasarkan jumlah *cluster* yang telah dihitung secara manual.

5.3.1 Skenario Uji Coba 1

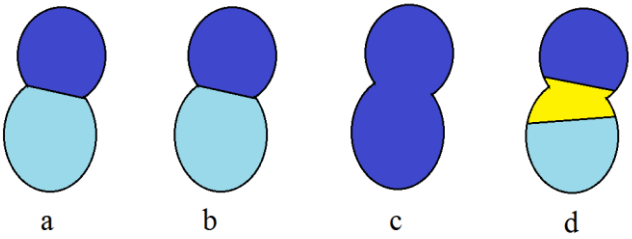
Pada skenario uji coba 1, akurasi hasil segmentasi dihitung dengan membandingkan metode yang digunakan dengan segmentasi manual. Segmentasi manual dilakukan dengan menandai / mewarnai area sel darah merah pada citra. Metriks untuk mengevaluasi akurasi segmentasi menggunakan Akurasi, *False Positive* (FP) *rat*e, dan *False Negative* (FN) *rate*. Akurasi dihitung sebagai rasio jumlah piksel yang diidentifikasi dengan benar (*True Positive* (TP)) dan jumlah piksel *background* (*True Negative* (TN)) terhadap jumlah piksel *ground truth*.

Nilai FP rate dihitung sebagai rasio jumlah piksel yang teridentifikasi sebagai objek oleh metode yang digunakan namun teridentifikasi sebagai *background* oleh citra *ground truth*.

Nilai FN rate dihitung sebagai rasio jumlah piksel yang teridentifikasi sebagai *background* oleh metode yang digunakan namun teridentifikasi sebagai objek oleh citra *ground truth*.

5.3.2 Skenario Uji Coba 2

Pada skenario uji coba 2,yaitu menghitung akurasi jumlah sel darah merah bertumpuk metode *correctly segmented, overgmentation* dan *undersegmentation* yang akan membandingkan hasil proses identifikasi sel darah merah bertumpuk dengan hasil penghitungan secara manual. Nilai *oversegmentation* dihitung berdasarkan jumlah *cluster* yang berlebih sedangkan *undersegmentation* dihitung berdasarkan jumlah *cluster* yang kurang dari seharusnya seperti yang ditunjukan pada Gambar 5.1.



Gambar 5.1 (a) Contoh objek sel mengandung *cluster* (b) *correctly segmented* (c) *undersegmentation* (d) *oversegmentation*

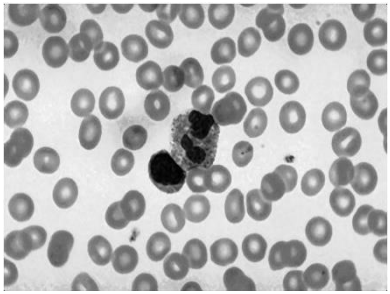
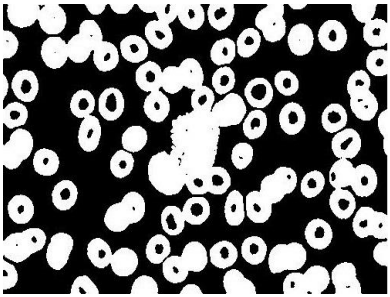
5.4 Hasil Uji Coba

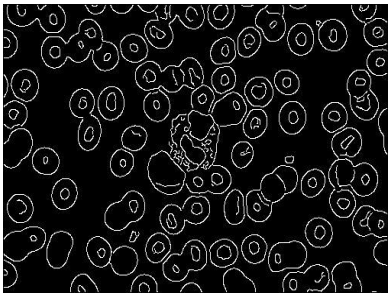
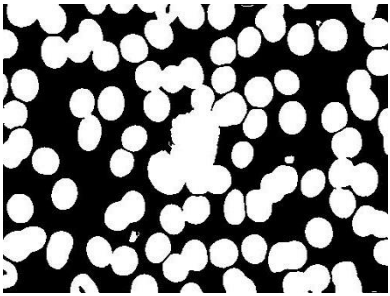
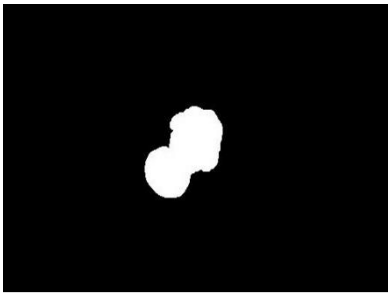
Pada subbab ini akan dijelaskan mengenai hasil uji coba program berdasarkan skenario uji coba yang telah dirancang pada subbab 5.3. Hasil skenario uji coba akan dijelaskan pada subbab 5.4.1 untuk skenario 1 dan subbab 5.4.2 untuk skenario 2.

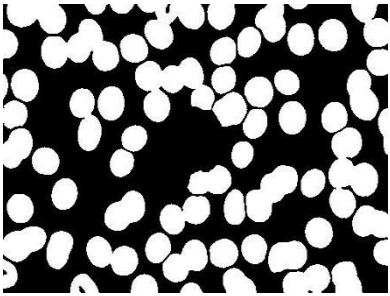
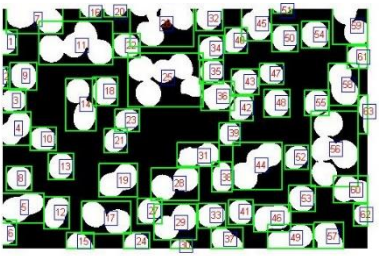
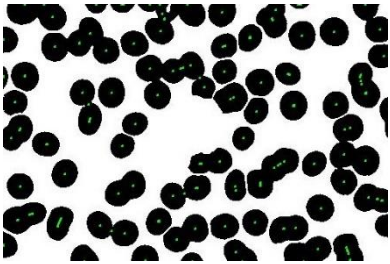
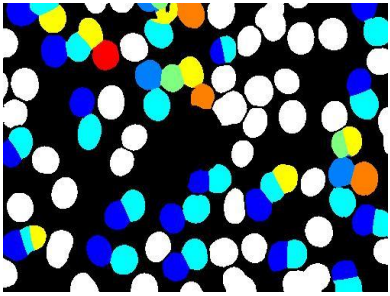
Hasil akhir uji coba dari pemrosesan identifikasi sel darah merah bertumpuk untuk setiap citra dapat dilihat pada Lampiran Uji Coba.

Hasil setiap proses identifikasi sel darah merah bertumpuk dari awal hingga akhir pada sampel citra ‘heme004’ ditunjukan pada Tabel 2.

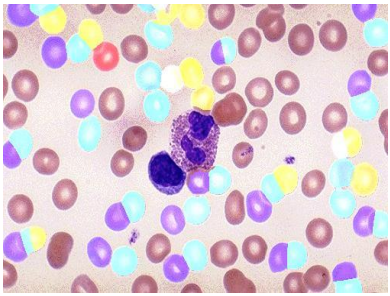
Tabel 2 Hasil pemrosesan data heme004

No.	Nama proses	Hasil
1	Input citra	
2	Median Filtering	
3	GrayThresholding	

4	Deteksi tepi canny	
5	Citra biner sel darah	
6	Citra biner sel darah putih	

7	Citra biner sel darah merah	
8	Penghitungan jumlah objek	
9	Penghitungan jumlah titik *cluster* awal	
10	Visualisasi pemisahan sel darah merah yang bertumpuk	

11 Citra gabungan



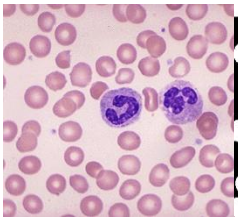
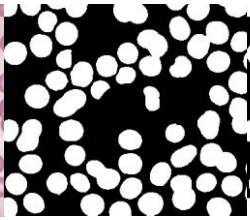
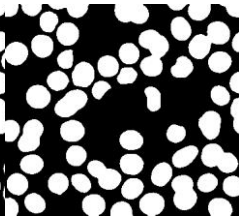
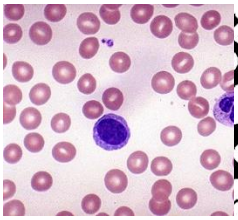
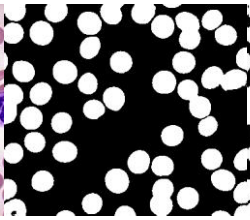
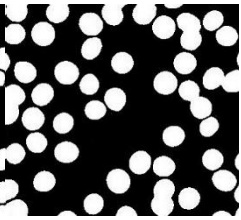
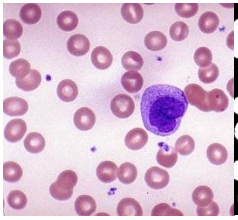
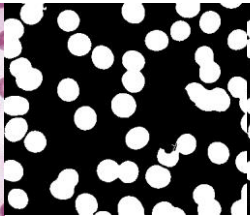
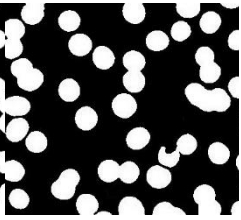
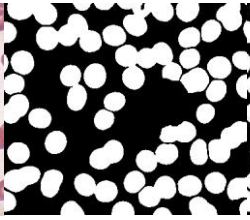
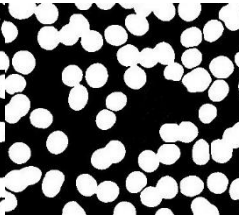
5.4.1 Hasil Uji Coba 1

Pada uji coba 1, dataset citra yang digunakan berjumlah 20 yaitu hasil segmentasi citra mikroskopik sel darah merah yang berjumlah 10 dan *ground truth* citra mikroskopik sel darah merah yang berjumlah 10. Hasil dari uji coba ini adalah Akurasi, *False Positive* (FP) *rat*e, dan *False Negative* (FN) *rate*. Pada Tabel 3 ditunjukan perbandingan dari citra RGB sel darah, hasil segmentasi sel darah merah dan *ground truth*. Hasil uji coba 1 yang berupa penghitungan akurasi, FP *rate* dan FN *rate* ditunjukan pada Tabel 4.

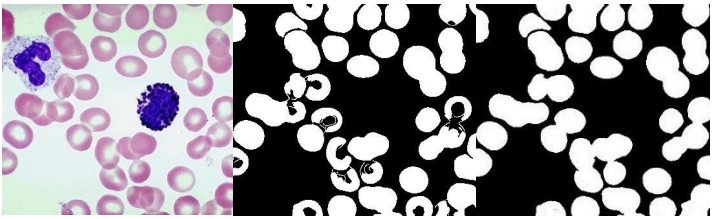
Tabel 3 Hasil akurasi uji coba 1

Citra	Akurasi (%)	FP rate (%)	FN rate (%)
heme001	98,28	1,27	0,45
heme002	96,82	1,81	1,37
heme003	97,55	1,48	0,97
heme004	97,63	1,55	0,82
heme005	93,83	1,96	4,21
heme084	97,32	1,73	0,95
mega2	97,24	2,16	0,60
mega3	97,70	1,42	0,88
mega4	97,64	1,52	0,84
rbc2	93,48	4,16	2,36
Rata – rata	96,75	1,34	1,91

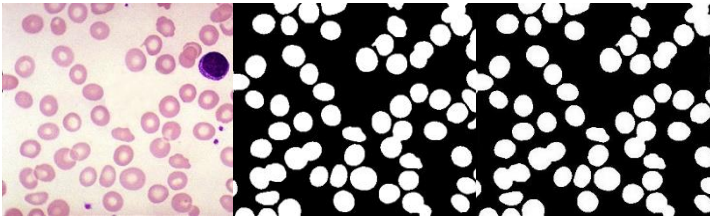
Tabel 4 Perbandingan citra

No.	Citra Asli	Hasil Segmentasi	Ground Truth
1			
2			
3			
4			

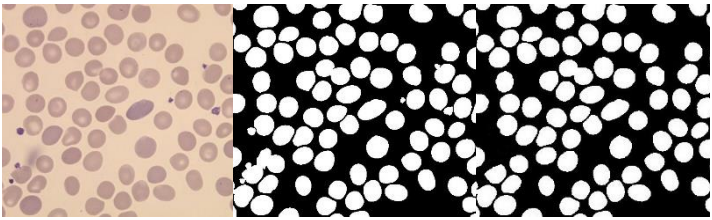
5



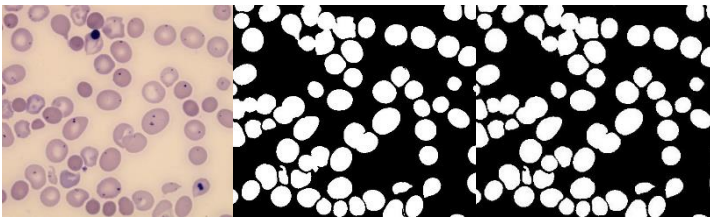
6



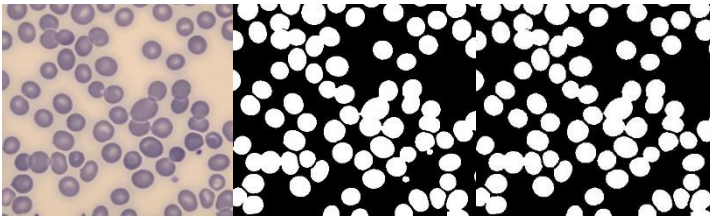
7

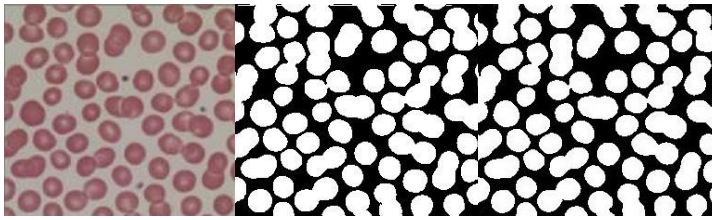


8



9



10 

Berdasarkan penghitungan yang telah dilakukan terhadap 10 citra uji, didapatkan rata-rata akurasi metode yang digunakan mencapai 96,75% dengan rata-rata error FP dan FN masing-masing sebesar 1,34% dan 1,91%.

5.4.2 Hasil Uji Coba 2

Pada uji coba 2, dataset citra yang digunakan berjumlah 10 yaitu citra mikroskopik sel darah Hasil dari uji coba ini adalah penghitungan jumlah sel darah merah tunggal dan bertumpuk yang dapat dilihat pada Tabel 5 dan akurasi penghitungan pada Tabel 6.

Tabel 5 Jumlah segmentasi sel darah merah

		Jumlah sel darah merah hasil metode		
Citra	**Jumlah sel darah merah**	**Correctly segmented**	**Underseg-mented**	**Overseg-mented**
heme001	75	69	3	3
heme002	66	60	1	5
heme003	52	45	3	4
heme004	63	53	6	4
heme005	42	31	3	8
heme084	76	69	0	7
mega2	95	86	0	9
mega3	68	62	1	5
mega4	76	70	0	6
rbc2	57	50	5	2

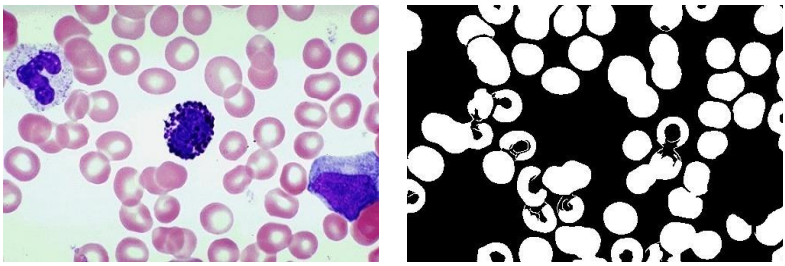
Tabel 6 Hasil akurasi uji coba 2

	Persentase (%)		
Citra	***Correctly segmented***	***Underseg-mented***	***Overseg-mented***
heme001	92	**4**	4
heme002	90,9	1,5	7,5
heme003	86,5	5,7	7,6
heme004	84,1	9,5	6,3
heme005	73,8	7,1	19
heme084	90,7	0	9
mega2	90,5	0	9
mega3	91,1	1,4	7
mega4	92,1	0	7
rbc2	87,7	8,7	3
Rata - rata	87,94	3,82	8,24

Berdasarkan penghitungan yang telah dilakukan terhadap 10 citra uji, didapatkan rata-rata akurasi (*correctly segmented*) sel darah merah bertumpuk mencapai 87,94% dengan rata-rata error *undersegmented* dan *oversegmented* masing-masing sebesar 3,82% dan 8,24%.

5.4.3 Analisis Hasil Uji Coba

Pada uji coba 1, akurasi pada citra ‘heme005’ memiliki akurasi error sebesar 4,21%, hal ini dikarenakan warna *concave* sel darah merah pada citra ‘heme005’ memiliki warna yang hampir sama dengan *background*, sehingga hal tersebut mempengaruhi hasil segmentasi, seperti yang ditunjukan pada Gambar 5.2.

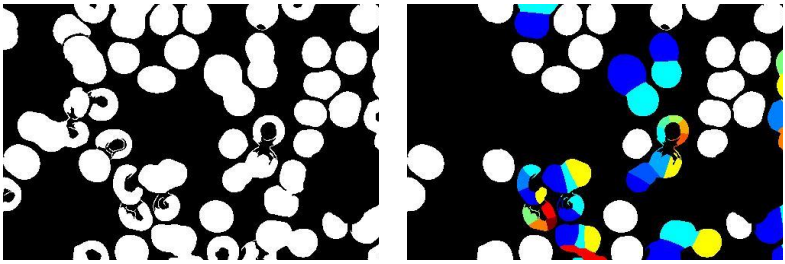


(a) **(b)**

Gambar 5.2 (a) citra asli heme005 (b) citra segmentasi heme005

Pada uji coba 2, akurasi pada data citra ‘heme005’ juga terbilang kecil yaitu sebesar 73,8%, hal ini disebabkan oleh dua hal yaitu :

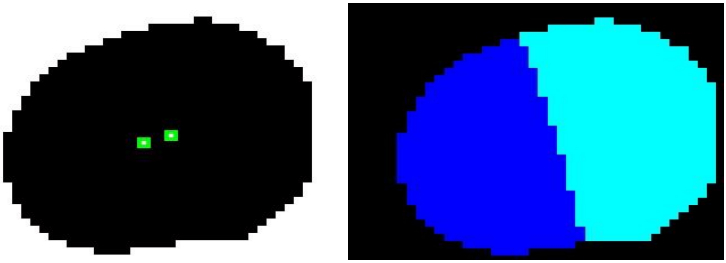
1. Hasil citra dari proses segmentasi yang kurang baik menyebabkan berkurangnya ketepatan dalam menentukan titik tengah *cluster* yang bertumpuk pada sel darah merah, seperti yang ditunjukan pada Gambar 5.3.



(a) **(b)**

Gambar 5.3 (a) citra segmentasi heme005 (b) citra visualisasi pemisahan sel darah merah bertumpuk heme005

2. Hasil dari pembatasan *cluster* pada metode *H-maxima* berpengaruh dalam penentuan batas *cluster* yang akan diproses, jika nilai variabel *depth* terlalu besar, maka akan mempengaruhi hasil dari pemisahan sel bertumpuk yang seringkali menyebabkan *oversegmented*, seperti yang dtunjukan pada Gambar 5.4.



(a) **(b)**

Gambar 5.4 (a) citra penentuan titik tengah *cluster* dengan H-maxima (b) citra visualisasi pemisahan sel darah merah bertumpuk

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini berisi kesimpulan mengenai pengujian dan evaluasi yang telah dilakukan. Selain itu, terdapat beberapa saran guna penyempurnaan ke depan.

6.1 Kesimpulan

Kesimpulan yang diperoleh berdasarkan pengujian dan evaluasi yang telah dilakukan adalah sebagai berikut:

1. Hasil uji coba 1 dari metode segmentasi yang digunakan dibandingkan dengan citra *ground truth* menghasilkan rata-rata akurasi mencapai 96,75% dengan rata-rata error FP dan FN masing-masing sebesar 1,34% dan 1,91%.
2. Metode *unsupervised Bayesian classification* dapat digunakan unuk mengidentifikasi sel darah merah yang bertumpuk karena pada uji coba 2 didapatkan rata-rata akurasi (*correctly segmented*) sel darah merah yang bertumpuk mencapai 87,94% dengan rata-rata error *undersegmented* dan *oversegmented* masing-masing sebesar 3,82% dan 8,24%.
3. Metode *masking* sel darah putih pada tahapan segmentasi dapat mempengaruhi akurasi dalam proses identifikasi sel darah merah bertumpuk.
4. Penggunaan metode *H-maxima* sebagai pembatasan nilai *cluster* sangat bergantung pada nilai variabel *depth*, jika nilai *depth* yang diberikan terlalu besar maka akan mempengaruhi bentuk pemisahan sel darah merah bertumpuk.

6.2 Saran

Saran yang hendak disampaikan terkait dengan Tugas Akhir ini, yang adalah diperlukannya peninjauan ulang kembali metode *H-maxima* dalam menentukan nilai batasan *cluster* yang akan dilakukan proses *validasi cluster*, dan juga dibutuhkan proses segmentasi yang lebih baik lagi jika citra mikroskopik sel darah merah memiliki warna *concave* yang sama dengan *background*.

DAFTAR PUSTAKA

[1] C. Jung, K. C, S. W. Chae and S. Oh, "Unsupervised Segmentation of Overlapped Nuclei Using Bayesian Classification," *IEEE Transactions on Biomedical Engineering*, vol. 57, no. 12, pp. 2825-2832, December 2010.

[2] F. Effendy, C. Fatichah and D. Purwitasari, "Identifikasi Sel Bertumpuk Berdasarkan Fitur Geometri Pada Sel Darah Merah," in *Prosiding Seminar Nasional Pascasarjana XIII – ITS*, Surabaya, 2013.

[3] M. Bouguessa, S. Wang and H. Sun, "An objective approach to cluster validation," *Pattern Recognition Letters*, vol. 27, no. 13, p. 1419–1430, 2006.

[4] R. C. Gonzales and R. E. Woods, Digital Image Processing, 2nd Edition ed., Prentice Hall, 2002.

[5] A. R. Douglas, "Gaussian Mixture Models," in *Encyclopedia of Biometrics*, Springer, 2009, pp. 659-663.

[6] R. O. Duda, P. E. Hart and D. G. Stork, Pattern Classification, 2nd Edition ed., Wiley-Intersceince, 2000.

[7] R. Fisher, S. Perkins, A. Walker and E. Wolfart, "The University of Edinburgh, school of informatics," [Online]. Available: http://homepages.inf.ed.ac.uk/rbf/HIPR2/distance.htm.

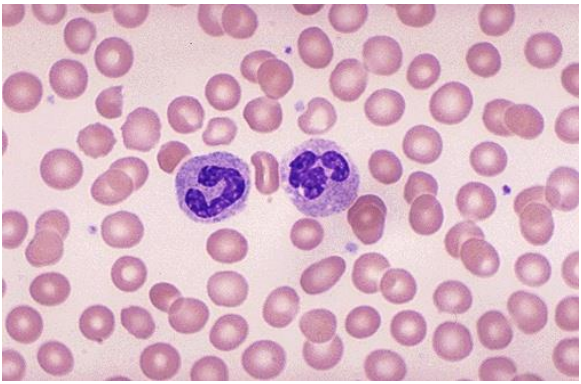
[8] "Regional Maximal (imregionalmax)," [Online]. Available: https://nf.nci.org.au/facilities/software/Matlab/toolbox/images/imregionalmax.html.

[9] A. P. Singh, "Carnegie Mellon School of Computer Science," November 2005. [Online]. Available: http://www.cs.cmu.edu/~awm/15781/assignments/.

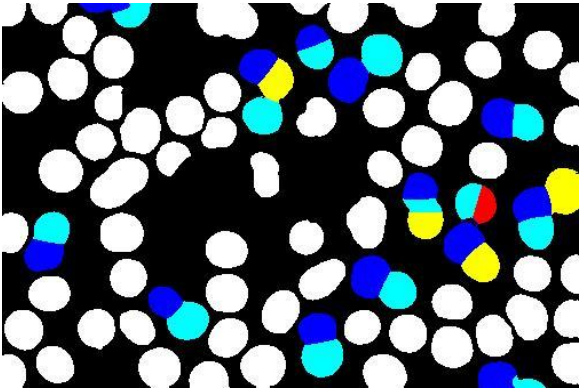
LAMPIRAN UJI COBA

Citra HEME001

Citra RGB

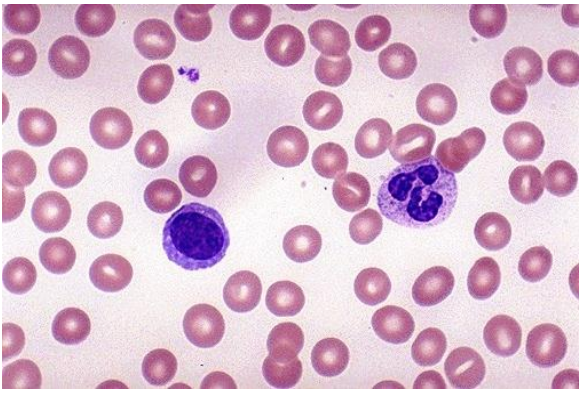


Citra visualisasi pemisahan sel darah merah yang bertumpuk

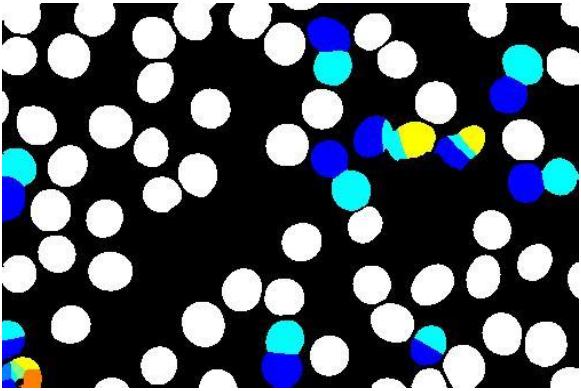


Citra HEME002

Citra RGB



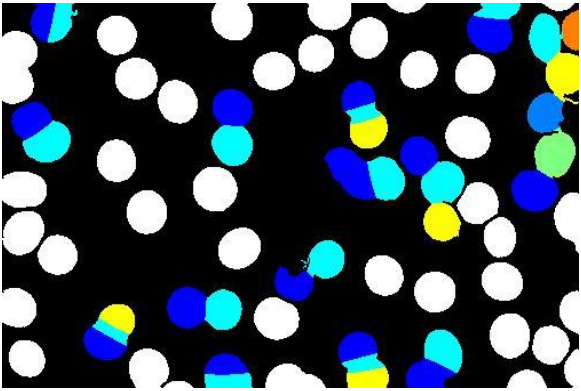
Citra visualisasi pemisahan sel darah merah yang bertumpuk



Citra HEME003

Citra RGB

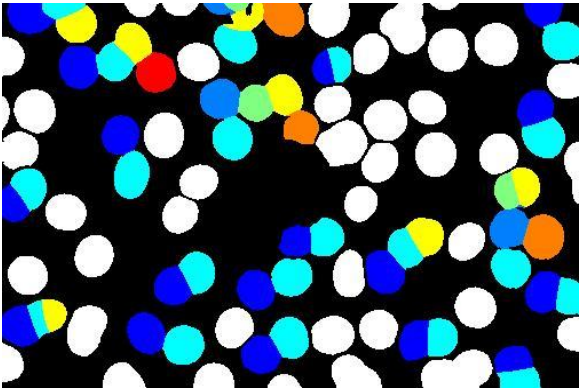
Citra visualisasi pemisahan sel darah merah yang bertumpuk



Citra HEME004

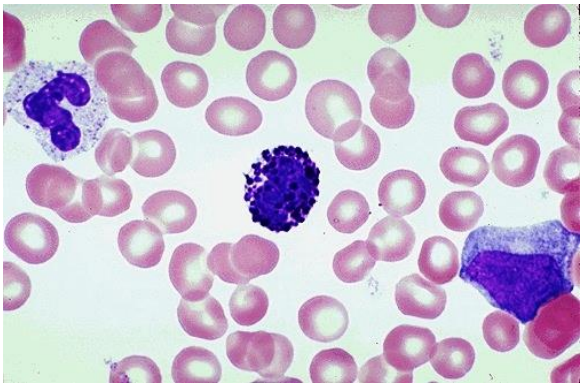
Citra RGB

Citra visualisasi pemisahan sel darah merah yang bertumpuk

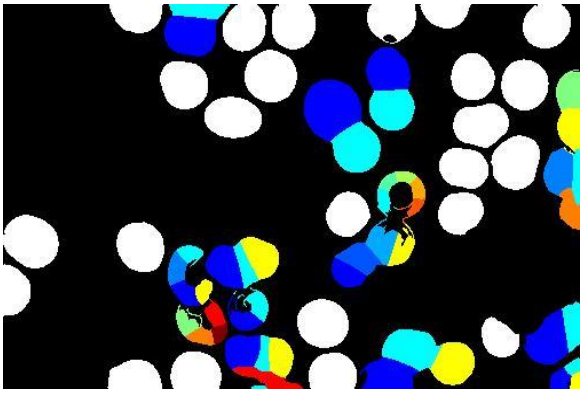


Citra HEME005

Citra RGB

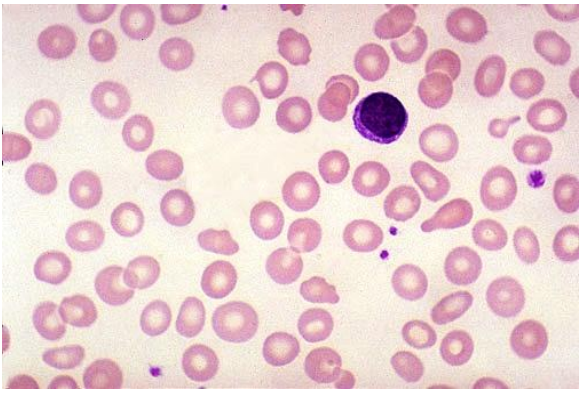


Citra visualisasi pemisahan sel darah merah yang bertumpuk

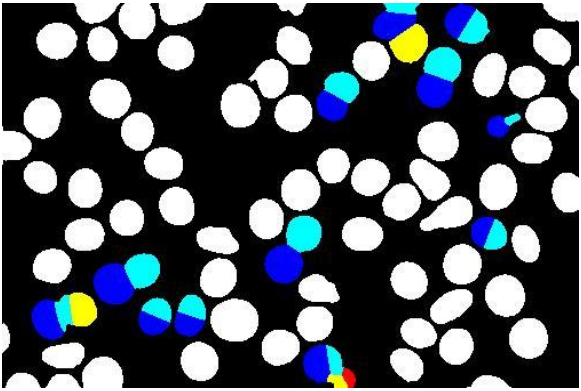


Citra HEME084

Citra RGB

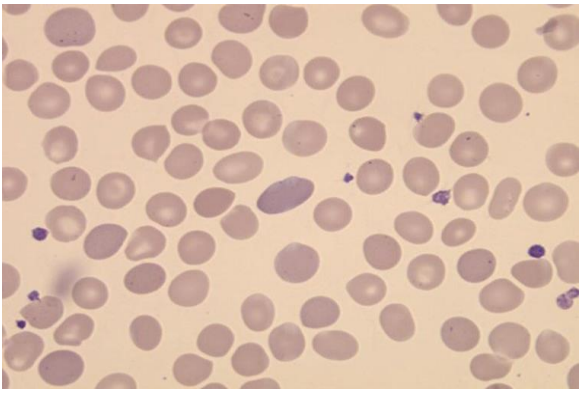


Citra visualisasi pemisahan sel darah merah yang bertumpuk

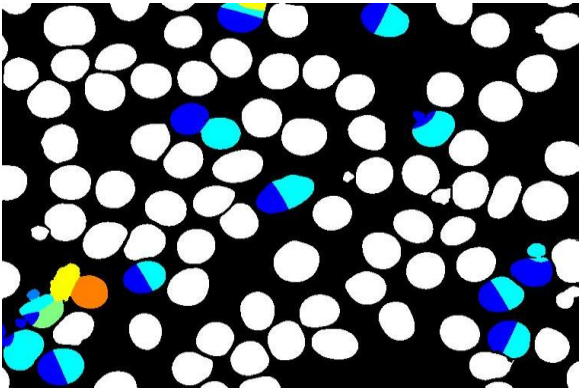


Citra MEGA2

Citra RGB

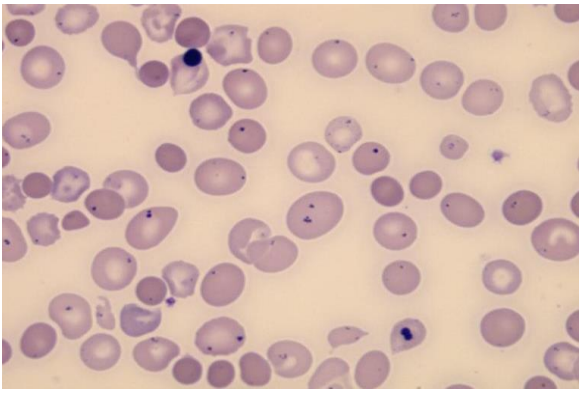


Citra visualisasi pemisahan sel darah merah yang bertumpuk

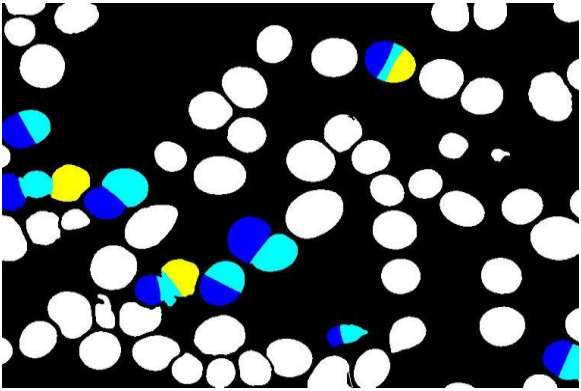


Citra MEGA3

Citra RGB

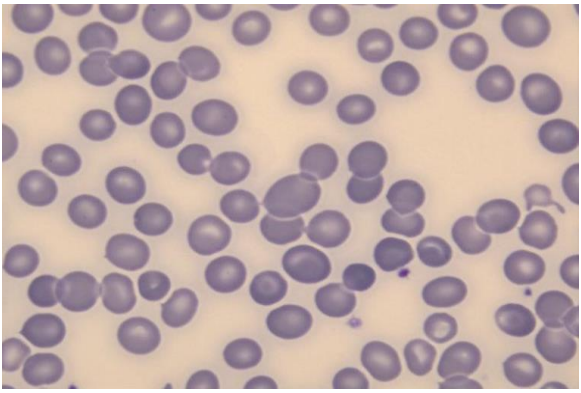


Citra visualisasi pemisahan sel darah merah yang bertumpuk

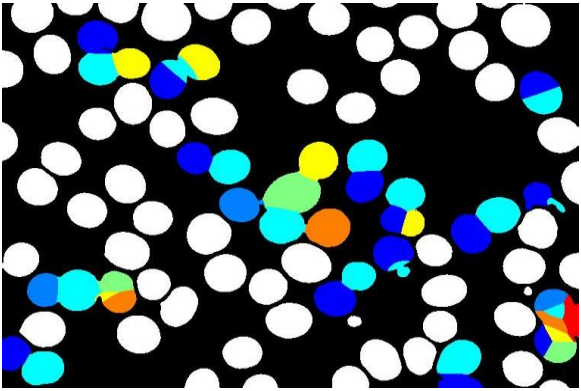


Citra MEGA4

Citra RGB

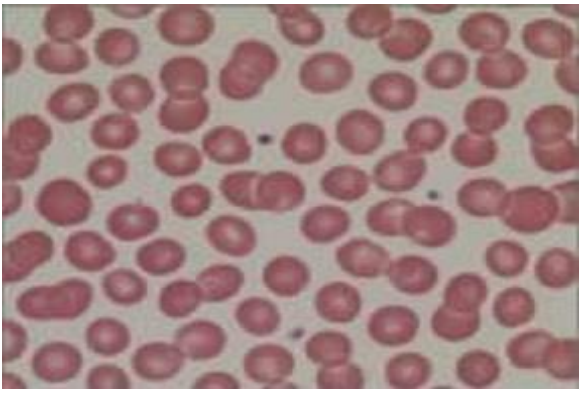


Citra visualisasi pemisahan sel darah merah yang bertumpuk

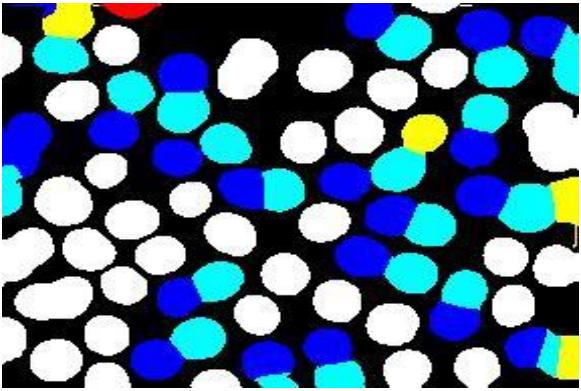


Citra rbc2

Citra RGB



Citra visualisasi pemisahan sel darah merah yang bertumpuk



BIODATA PENULIS

RM Reza Rizky Pratama, merupakan anak pertama dari tiga bersaudara dan biasa dipanggil Reza. Penulis dilahirkan di Surabaya pada tanggal 8 Desember 1992. Penulis telah menempuh pendidikan formal di TK Tadika Puri (1996-1998), SD Negeri Dr. Soetomo V Surabaya (1998-2004), SMP Negeri 6 Surabaya (2004-2007), SMA Negeri 2 Surabaya (2007-2010). Pada tahun 2010 penulis diterima di S1 Jurusan Teknik Informatika Institut Teknologi Sepuluh Nopember Surabaya melalui jalur SNMPTN dan terdaftar dengan NRP 5110100181. Di Jurusan Teknik Informatika, penulis mengambil bidang minat Komputasi Cerdas dan Visualisasi (KCV). Selain bidang akademik, penulis suka menghabiskan waktu untuk berolahraga, mendengarkan musik dan bermain game. Penulis dapat dihubungi melalui alamat email rm.rezarp@gmail.com.