Multilevel Image Thresholding Memanfaatkan Firefly Algorithm, Improved Bat Algorithm, dan Symbiotic Organisms Search

C. Pickerling, *Informatika Institut Sains dan Teknologi Terpadu Surabaya*,
Hendrawan Armanto, *Informatika Institut Sains dan Teknologi Terpadu Surabaya*,
dan Stefanus Kurniawan B., *Informatika Institut Sains dan Teknologi Terpadu Surabaya*

***Abstrak* — Multilevel image thresholding adalah teknik penting dalam pemrosesan gambar yang digunakan sebagai dasar image segmentation dan teknik pemrosesan tingkat tinggi lainnya. Akan tetapi, waktu yang dibutuhkan untuk pencarian bertambah secara eksponensial setara dengan banyaknya threshold yang diinginkan. Algoritma metaheuristic dikenal sebagai metode optimal untuk memecahkan masalah perhitungan yang rumit. Seiring dengan berkembangnya algoritma metaheuristic untuk memecahkan masalah perhitungan, penelitian ini menggunakan tiga algoritma metaheuristic, yaitu Firefly Algorithm (FA), Symbiotic Organisms Search (SOS), dan Improved Bat Algorithm (IBA). Penelitian ini menganalisis solusi optimal yang didapatkan dari percobaan masing-masing algoritma. Hasil uji coba masing-masing algoritma saling dibandingkan untuk menentukan kelemahan dan kelebihan setiap algoritma berdasarkan performanya. Hasil uji coba menyatakan tiga algoritma tersebut memiliki performa berbeda dalam optimisasi multilevel image thresholding.**

***Kata Kunci* — Multilevel Image Thresholding, Threshold, Metaheuristic**

I. PENDAHULUAN

Dewasa ini, kebutuhan untuk pengolahan gambar semakin berkembang seiring dengan kemajuan teknologi. Pengolahan ini memegang peran yang penting dalam analisis dan interpretasi gambar di berbagai aspek kehidupan seperti medis, navigasi, deteksi kejadian secara otomatis, pengamatan, pengenalan tekstur beserta pola dan deteksi kecacatan. Pengembangan teknik citra digital dan teknologi komputasi membawa kemajuan pada ilmu pencitraan.

Salah satu teknik penting dalam ilmu pencitraan adalah image segmentation. Image segmentation membagi gambar menjadi beberapa kelompok dengan kemiripan sifat seperti intensitas, warna, dan kontur. Biasanya, image segmentation melambangkan tahapan awal pemahaman gambar dan dilanjutkan ke metode tingkat lanjut seperti ekstraksi fitur, interpretasi semantik, pengenalan gambar, dan klasifikasi benda.

Multilevel image thresholding merupakan teknik umum dalam menjalankan image segmentation. Gambar yang dihasilkan gadget menghasilkan resolusi yang berbeda sesuai spesifikasi kamera yang dipakai. Kamera dengan resolusi tinggi dapat menghasilkan gambar dengan ukuran besar. Multilevel image thresholding membutuhkan waktu yang lama dan juga banyak perhitungan dikarenakan besarnya ukuran gambar dan banyaknya threshold.

Masalah waktu dan kebutuhan perhitungan yang tinggi akan menyebabkan proses pengolahan gambar tidak efisien. Untuk mengatasi masalah ini, algoritma metahuristic diharapkan mampu mempercepat konvergensi dan mengurangi waktu perhitungan. Sebagian algoritma metaheuristic berasal dari perilaku makhluk hidup dan penerapan sistem di alam. Algoritma metaheuristic yang mendapat inspirasi dari makhluk hidup seperti Firefly Algorithm, Improved Bat Algorithm, dan Symbiotic Organisms Search adalah algoritma yang belum banyak diterapkan untuk penyelesaian multilevel image thresholding. Pada penelitian ini akan dibahas implementasi Firefly Algorithm, Improved Bat Algorithm, dan Symbiotic Organisms Search untuk multilevel image thresholding

II. TINJAUAN PUSTAKA

Penelitian ini menggunakan beberapa tinjauan pustaka sebagai dasar agar penelitian dapat berjalan sesuai arah yang tepat. Tinjauan pustaka yang digunakan antara lain:

A. *OTSU Thresholding [1]*

Teknik thresholding melakukan pemrosesan gambar berdasarkan informasi yang terdapat pada histogram *gray-level*. Jika input gambar *grayscale* I dianggap sebagai sekumpulan pixel A, maka multilevel thresholding dapat didefinisikan sebagai metode untuk membagi kumpulan pixel A menjadi kelompok-kelompok kecil (A_0 , A_1, …, A_n) dengan kumpulan threshold (t_0, t_1, …, $t_{(n-1)}$) dengan definisi berikut:

$$\begin{aligned} A_0 &= \{x: 0 \le f(x) < t_0\}, \\ A_1 &= \{x: t_0 \le f(x) < t_1\}, \\ &\vdots \\ A_n &= \{x: t_{n-1} \le f(x) \le L-1\}. \end{aligned} \quad (1)$$

C. Pickerling, Departemen Informatika, Institut Sains dan Teknologi Terpadu Surabaya, Jawa Timur, Indonesia (e-mail: pickerling@stts.edu)
Hendrawan Armanto, Departemen Informatika, Institut Sains dan Teknologi Terpadu Surabaya, Jawa Timur, Indonesia (e-mail: hendrawan@stts.edu)
Stefanus Kurniawan B., Departemen Informatika, Informatika, Institut Sains dan Teknologi Terpadu Surabaya, Surabaya, Jawa Timur, Indonesia.

$x = (x_1, x_2)$ adalah pixel yang didefinisikan dengan koordinat x_1 dan x_2 pada sistem koordinat Kartesius, f(x) melambangkan nilai gray-level untuk pixel x, dimana nilai f(x) berada di rentang [0, …, 255]. Tujuan dari multilevel thresholding adalah menghitung nilai optimal untuk kumpulan threshold $(t_0, t_1, \ldots, t_{(n-1)})$. Kelompok-kelompok kecil $(A_0, A_1, \ldots, A_n)$ melambangkan bagian berbeda pada gambar I. Berlaku $A_i \cap A_j = \emptyset$ untuk setiap kumpulan pixel A, dan gabungan seluruh kelompok kecil adalah keseluruhan input gambar grayscale I.

Pemilihan threshold optimal untuk image thresholding semula tidak memiliki banyak perhitungan, lain halnya dengan multilevel image thresholding yang memiliki banyak perhitungan dan membutuhkan waktu yang lama seiring bertambahnya threshold untuk dioptimalkan. Kumpulan threshold optimal untuk multilevel image thresholding ditentukan dengan mengoptimalkan kriteria sesuai histogram gray-level dari input gambar grayscale I. [2]

Metode Otsu dalam memaksimalkan variansi between-class adalah metode populer untuk image thresholding. Algoritma untuk metode ini dijelaskan sebagai berikut. Diasumsikan gambar grayscale I memiliki kelas sebanyak m+1 $[C_0, C_1, \ldots, C_m]$, di mana m adalah inputan dimensi dan probabilitas setiap kelas dengan definisi berikut: [3]

$$\begin{aligned} \omega_0 &= \sum_{i=0}^{t_0-1} p_i, \\ \omega_1 &= \sum_{i=t_0}^{t_1-1} p_i, \\ &\vdots \\ \omega_m &= \sum_{i=t_m-1}^{L-1} p_i, \end{aligned} \tag{2}$$

di mana t_{number} adalah threshold yang memisahkan gambar I menjadi beberapa kelas. Untuk threshold sebanyak m yang membagi gambar I menjadi (m+1) kelas, dihitung nilai fungsi objektif dengan memaksimalkan variansi between-class seperti berikut:

$$\begin{aligned} Maximize\ J(t) &= \sigma_0 + \sigma_1 + \cdots + \sigma_m, \\ \sigma_0 &= \omega_0(\mu_0 - \mu_T)^2, \\ \sigma_1 &= \omega_1(\mu_1 - \mu_T)^2, \\ &\vdots \\ \sigma_m &= \omega_m(\mu_m - \mu_T)^2. \end{aligned}$$

B. *Firefly Algorithm [4]*

Firefly Algorithm (FA) adalah algoritma metaheuristic dengan inspirasi perilaku kunang-kunang dalam memancarkan cahaya. Pada Firefly Algorithm, terdapat tiga macam aturan yang sudah dianggap ideal: (1) semua kunang-kunang dianggap unisex, jadi seekor kunang-kunang dapat tertarik dengan kunang-kunang yang lain tanpa mempedulikan jenis kelamin kunang-kunang lain; (2) ketertarikan kunang-kunang pada kumpulan berbanding lurus dengan tingkat terangnya cahaya. Untuk dua kunang-kunang yang saling memancarkan cahayanya, kunang-kunang dengan cahaya yang lebih redup akan bergerak ke arah kunang-kunang yang lebih terang. Apabila jarak mereka semakin jauh, ketertarikan mereka akan semakin berkurang. Seekor kunang-kunang akan bergerak secara random apabila tidak ada yang lebih terang daripada kunang-kunang tersebut; (3) terangnya kunang-kunang dipengaruhi atau ditentukan oleh bagaimana fungsi objektifnya.

Pergerakan kunang-kunang i yang tertarik oleh kunang-kunang j dengan pancaran cahaya lebih terang ditentukan oleh:

$$\begin{aligned} x_i &= x_i + \beta(x_j - x_i) + \alpha\left(rand - \frac{1}{2}\right), \\ \beta(d_{ij}) &= \beta_0 e^{-\gamma {d_{ij}}^2}, \end{aligned} \tag{4}$$

dengan x_i adalah posisi kunang-kunang i dan x_j adalah posisi kunang-kunang j. $\beta(d_{ij})$ melambangkan ketertarikan kunang-kunang i terhadap kunang-kunang j, di mana d_{ij} adalah jarak dari kunang-kunang i dan kunang-kunang j. β_0 adalah ketertarikan pada $d_{ij}=0$, dan γ adalah koefisien penyerapan cahaya. α adalah parameter yang bernilai random, dan rand adalah nilai random antara [0, 1] yang dirandom setiap kali dipakai.

Langkah utama Firefly Algorithm dapat dituliskan sebagai berikut: (1) menghitung nilai fungsi objektif setiap kunang-kunang; (2) kunang-kunang dengan nilai fungsi objektif yang kurang optimal bergerak menuju kunang-kunang yang lebih optimal; (3) menentukan kunang-kunang yang memiliki nilai fungsi objektif paling optimal atau terbaik dan mencatatnya; (4) mengulangi iterasi hingga batas iterasi maksimum; (5) penggunaan kunang-kunang terbaik pada akhir iterasi maksimum.

C. *Improve Bat Algorithm [5]*

Bat Algorithm dipilih untuk menyelesaikan masalah multilevel image thresholding karena mudah untuk diimplementasikan dan menghasilkan solusi yang baik untuk threshold sebanyak m dimensi, di mana m adalah angka yang kecil. Akan tetapi, Bat Algorithm dianggap kurang berhasil untuk threshold berdimensi besar, sehingga perlu ada perubahan pada algoritma ini [7]. Improved Bat Algorithm (IBA) dibuat untuk mengatasi kekurangan dari Bat Algorithm.

Improved Bat Algorithm menggabungkan dua perhitungan untuk pencarian solusi dari Bat Algorithm dan algoritma Differential Evolution (DE). Improved Bat Algorithm mengambil operator mutation dan crossover dari algoritma Differential Evolution dengan tujuan mempercepat pencarian solusi secara konvergen dan mencapai keseimbangan yang baik antara peningkatan dan penggolongan. Operator mutation dan crossover digunakan untuk meningkatkan perhitungan solusi lain (x_{new}) dari Bat Algorithm semula, sehingga Improved Bat Algorithm dapat mengeksplorasi dan mengeksploitasi area pencarian baru dan dapat menghindari jebakan local optima.

Pada Bat Algorithm, eksplorasi dan eksploitasi diatur oleh besarnya pulse emission rate. Dengan menganalisa perhitungan ini, algoritma semakin kehilangan kemampuan eksplorasinya seiring berjalannya iterasi. Pada Improved Bat

Algorithm, bagian ini dimodifikasi agar dapat bergantian dalam eksplorasi dan eksploitasi lebih baik. Kemampuan eksplorasi dari Bat Algorithm dimodifikasi dengan menambahkan operator crossover dan mutation.

Meskipun modifikasi yang telah dijelaskan di atas dapat meningkatkan performa banyak solusi, beberapa solusi akan tetap terjebak di dalam local optima. Untuk memperbaiki kekurangan pada modifikasi sebelumnya, ditambahkan modifikasi kedua yang mendapatkan inspirasi dari lebah pengintai pada fase mengintai milik algoritma Artificial Bee Colony (ABC). Ketika beberapa solusi terjebak pada local optima setelah beberapa iterasi, akan ada perhitungan yang memantau apakah jumlahnya melebihi batas. Ketika batas tersebut dilalui, solusi akan mencari area baru agar dapat terlepas dari local optima.

Dengan adanya penggunaan echolocation dari microbats, dibuatlah Bat Algorithm yang terinspirasi dari perilaku kelelawar tersebut. Improved Bat Algorithm memiliki peraturan yang sama dengan Bat Algorithm. Untuk menjalankan Improved Bat Algorithm, berikut ini adalah peraturan yang sudah dianggap ideal: (1) semua kelelawar menggunakan echolocation untuk memperkirakan jarak, dan mereka dianggap sudah mengetahui sekitarnya; (2) kelelawar terbang secara random dengan kecepatan v_i pada posisi x_i dengan frekuensi tetap f_{min}, panjang gelombang λ yang bervariasi, dan kebisingan A_0 untuk mencari mangsa. Mereka dapat menyesuaikan panjang gelombang yang dipancarkan secara otomatis dan mengatur pulse emission rate r [0, 1] tergantung dekatnya mereka dengan mangsa; (3) meskipun kebisingan dapat bervariasi dengan banyak cara, diasumsikan bahwa kebisingan bervariasi dari nilai positif A_0 hingga nilai minimum A_{min} yang konstan.

Pada Improved Bat Algorithm, populasi kelelawar melambangkan keseluruhan solusi. Detail langkah-langkah Improved Bat Algorithm dijelaskan sebagai berikut:

Langkah 1 (inisialisasi populasi kelelawar). Improved Bat Algorithm memunculkan populasi awal secara random dengan ukuran populasi kelelawar N (i=1, 2, …, N). N juga disebut sebagai banyaknya solusi, di mana setiap solusi memiliki m dimensi. Pada tahapan inisialisasi ini, dibuat parameter yang memberikan batasan kelelawar agar tidak terjebak pada local optima yang berkaitan dengan langkah 4. Selain itu perlu adanya input awal berupa nilai kebisingan awal (A_i^0), pulse emission rate awal (r_i^0), termasuk tambahan parameter Differential Evolution berupa bobot differential (F) dan kemungkinan crossover (C_r). Fungsi objektif untuk semua solusi dievaluasi dan iterasi untuk langkah berikutnya dapat dimulai. Improved Bat Algorithm mendeteksi solusi terbaik sebagai x_{best} sebelum iterasi dapat dimulai.

Langkah 2 (menghitung populasi baru). Solusi baru dihitung agar kelelawar dapat bergerak. Untuk iterasi saat ini (t), perhitungan solusi baru (x_i^t) dilakukan dengan menggerakkan solusi lama dari iterasi sebelumnya (x_i^{t-1}) dengan kecepatan (v_i^t) dan frekuensi (f_i^t) seperti berikut:

$$\begin{aligned} x_i^t &= x_i^{t-1} + v_i^t, \\ v_i^t &= v_i^{t-1} + (x_i^t - x_{best}) * f_i^t, \\ f_i^t &= f_{min} + (f_{max} - f_{min}) * \beta. \end{aligned} \tag{5}$$

Pada langkah perhitungan ini, Improved Bat Algorithm memantau kondisi perbatasan dengan solusi baru (x_i^t). Apabila nilai suatu variabel keluar dari batasan, maka nilai variabel tersebut perlu dirubah.

Langkah 3 (meningkatkan solusi terbaik saat ini dengan operator DE). Untuk setiap solusi baru (x_i^t), dibuatlah solusi lain (x_{new}) seperti berikut:

$$\begin{aligned} x_{new} &= \begin{cases} x_{dif}^t, & if\ rand_1 > r_i^t, \\ x_{loc}^t, & else, \end{cases} \\ r_i^t &= r_i^0\left(1 - e^{-\beta t}\right), \end{aligned} \tag{6}$$

di mana $rand_1$ adalah nilai random dari rentang [0, 1], r_i^t adalah pulse emission rate. x_{dif}^t adalah tambahan modifikasi berprinsip Differential Evolution (DE) dengan adanya operator untuk mutation dan crossover, sedangkan x_{loc}^t adalah operator pencarian lokal yang ada dari Bat Algorithm. Operator mutation dan crossover dituliskan sebagai berikut:

$$x_{dif,j}^t = \begin{cases} x_{c,j}^t + F\left(x_{a,j}^t - x_{b,j}^t\right), & if\ (rand_2 < C_r\ or\ j = j_r), \\ x_{i,j}^t, & else, \end{cases} \tag{7}$$

di mana $x_{a,j}^t$, $x_{b,j}^t$, $x_{c,j}^t$ adalah nilai random dari [0, N-1] yang berada pada iterasi kini (t) di posisi dimensi j, F adalah bobot differential yang mengatur besarnya modifikasi, C_r adalah kemungkinan crossover, j_r adalah dimensi bernilai random, dan $rand_2$ merupakan nilai random dari rentang [0, 1]. Penambahan operator differential $x_{dif,j}^t$ berguna untuk meningkatkan keberagaman posisi kelelawar dan mencapai akurasi serta efisiensi pencarian. Operator pencarian lokal dituliskan sebagai berikut:

$$\begin{aligned} x_{loc,j}^t &= \begin{cases} x_{lbest,j}^t, & if\ (f(x_{lbest,j}^t) > f(x_{i,j}^t))), \\ x_{i,j}^t, & else, \end{cases} \\ x_{lbest,j}^t &= x_{best,j}^{t-1} + \epsilon A_{i,j}^{t-1}. \end{aligned} \tag{8}$$

Langkah 4 (mengecek solusi baru). Pada langkah ini, solusi lain (x_{new}) dapat diterima sebagai solusi baru dan $f(x_{new})$ sebagai nilai fungsi objektif dengan detail:

$$\begin{aligned} &\left(x_i^t, f(x_i^t)\right) \\ &= \begin{cases} \left(x_{new}, f(x_{new})\right), \\ if\ \left(rand_3 < A_i^t\ and\ f\left(x_{new}\right) > f\left(x_{i\_old}^{t-1}\right)\right), \\ \left(x_i^{t-1}, f\left(x_i^{t-1}\right)\right) tr_i = tr_i + 1, else, \end{cases} \\ &\qquad A_i^t = \alpha A_i^{t-1}, \end{aligned} \tag{9}$$

di mana $rand_3$ adalah nilai random dari rentang [0, 1]. tr_i adalah parameter tambahan agar solusi dapat keluar dari local optima dengan mengubah nilai x_i^t dengan nilai random. Apabila tr_i melebihi batas yang telah ditentukan, nilai x_i^t digantikan nilai random dan tr_i kembali ke nilai 0.

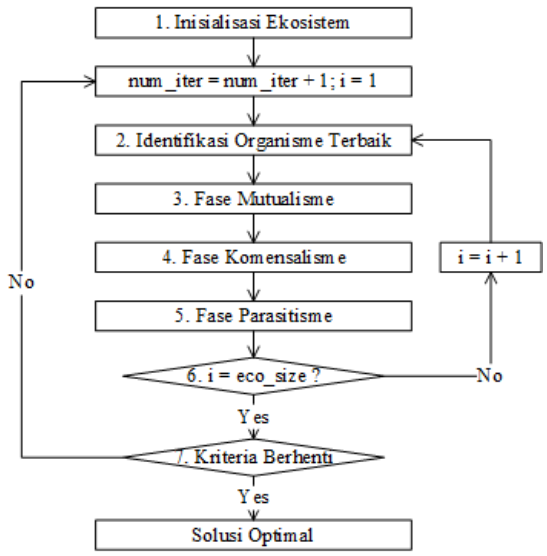
Langkah 5 (mengingat solusi terbaik). Mencatat solusi

terbaik saat ini (x_{best}) yang merupakan solusi dengan nilai fungsi objektif paling optimal.

Langkah 6 (mengecek iterasi). Apabila iterasi saat ini sama dengan iterasi maksimum, algoritma berhenti. Selain itu, nilai iterasi ditambahkan satu dan langkah 2 diulang kembali.

D. *Symbiotic Organisms Search [6]*

Mirip dengan algoritma metaheuristic lain yang berbasis populasi, Symbiotic Organisms Search menggunakan sebuah populasi yang terdiri dari kandidat solusi di dalam area pencarian untuk mendapatkan solusi terbaik secara keseluruhan. Symbiotic Organisms Search dimulai dengan inisialisasi populasi yang disebut dengan ekosistem. Pada ekosistem awal, kumpulan organisme dimunculkan dengan posisi acak di area pencarian. Setiap organisme merepresentasikan satu kandidat solusi untuk masalah yang ingin diselesaikan. Setiap organisme di dalam ekosistem memiliki nilai fungsi objektif yang menandakan derajat adaptasi pada objektif yang diinginkan.



Gambar. 1. Flowchart Symbiotic Organisms Search

Karakter pada setiap interaksi menentukan prinsip utama untuk setiap fase di dalam algoritma ini. Interaksi yang menguntungkan dua belah pihak berada di dalam fase mutualisme, menguntungkan satu pihak saja dan tidak mempengaruhi pihak lain berada di dalam fase komensalisme, menguntungkan satu pihak dan merugikan pihak lain berada di dalam fase parasitisme. Gambar 1 adalah flowchart dari penjelasan yang telah diberikan.

Fase Mutualisme. Contoh dari mutualisme yang menguntungkan dua belah pihak organisme adalah hubungan antara lebah dan tanaman berbunga. Lebah mendapatkan nectar dan tanaman berbunga mendapatkan penyerbukan. Fase Symbiotic Organisms Search ini menirukan hubungan saling menguntungkan tersebut. Di dalam Symbiotic Organisms Search, X_i adalah organisme yang merupakan anggota ke-i dari ekosistem. Organisme lain X_j dipilih secara random dari ekosistem agar dapat berinteraksi dengan X_i. Dua organisme menjalin hubungan saling menguntungkan dengan tujuan meningkatkan faktor bertahan hidup di dalam ekosistem. Kandidat solusi baru untuk X_i dan X_j dihitung berdasarkan hubungan saling menguntungkan di antara keduanya seperti berikut:

$$\begin{aligned} X_{i\_new} &= X_i + rand * (X_{best} - MV * BF_1), \\ X_{j\_new} &= X_j + rand * (X_{best} - MV * BF_2), \\ MV &= \frac{X_i + X_j}{2}. \end{aligned} \quad (10)$$

Variabel rand adalah nilai random berkisar [0, 1]. Peran dari parameter BF_1 dan BF_2 dijelaskan sebagai berikut. Pada hubungan mutualisme di dalam alam, satu organisme mungkin lebih diuntungkan daripada pihak organisme yang lain. Faktor keuntungan atau Benefit Factors (BF_1 dan BF_2) ditentukan secara random apakah bernilai 1 atau 2. Faktor ini merepresentasikan seberapa besar organisme diuntungkan. Mutual_Vector (MV) merupakan representasi karakteristik hubungan organisme X_i dan X_j. Bagian perhitungan (X_{best} - MV * BF) merefleksikan usaha bersama mereka untuk meraih keuntungan dalam bertahan hidup. Solusi baru dengan nilai yang lebih baik akan menggantikan solusi lama, selain itu solusi lama tetap bertahan.

Fase Komensalisme. Contoh dari komensalisme adalah hubungan ikan remora dan hiu. Ikan remora menempelkan bagian atas tubuhnya pada hiu. Ikan remora memakan sisa makanan yang ada, sehingga ikan ini mendapatkan keuntungan. Hiu tidak terpengaruh dengan kegiatan ikan remora yang menempel padanya. Mirip dengan fase mutualisme, organisme X_j dipilih secara random dari ekosistem untuk berinteraksi dengan organisme X_i. Pada keadaan ini, organisme X_i berusaha untuk mendapatkan keuntungan dari interaksi. Akan tetapi, organisme X_j tidak mendapatkan keuntungan dan tidak mendapat kerugian dari hubungan ini. Kandidat solusi baru didapatkan dari modifikasi organisme X_i yang dihitung berdasarkan simbiosis komensalisme antara organisme X_i dan X_j seperti berikut:

$$X_{i\_new} = X_i + rand * (X_{best} - X_j), \quad (11)$$

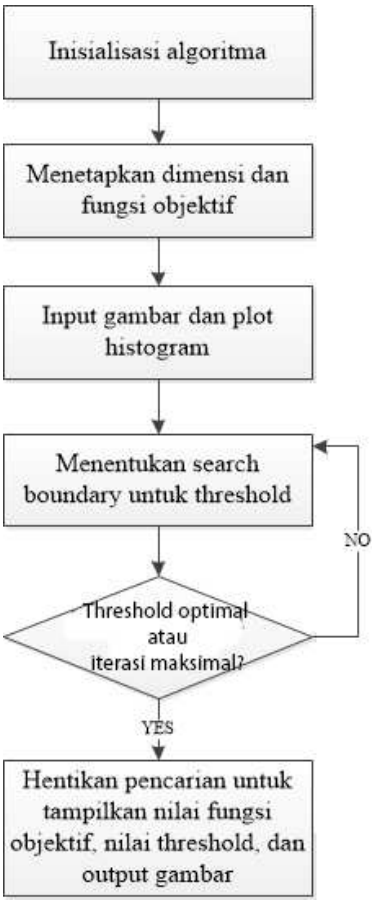
dengan bagian (X_{best} - X_j) yang merupakan perhitungan tentang berapa besar keuntungan yang disediakan oleh X_j untuk membantu X_i dalam meningkatkan kesuksesan bertahan hidup di dalam ekosistem menuju adaptasi terbaik yang dimiliki organisme X_{best}. Solusi baru dengan nilai yang lebih baik akan menggantikan solusi lama, selain itu solusi lama tetap bertahan.

Fase Parasitisme. Contoh dari parasitisme adalah parasit plasmodium yang biasanya menggunakan perantara nyamuk anopheles untuk berpindah-pindah ke tubuh manusia yang lain sebagai inang. Ketika parasit plasmodium tiba, tubuh manusia akan terserang penyakit malaria dan dapat mengakibatkan kematian sebagai akibatnya. Hal ini tentu merugikan manusia dan hanya menguntungkan parasit plasmodium yang memanfaatkan tubuh manusia. Pada Symbiotic Organisms Search, organisme X_i memiliki peran yang serupa dengan nyamuk anopheles sebagai media

penyebaran parasit bernama Parasite_Vector. Parasite_Vector dibentuk di dalam area pencarian solusi dengan menduplikasi organisme X_i, kemudian memodifikasi hasil duplikasi organisme X_i. Modifikasi ini mengganti solusi yang dimiliki induk pada dimensi yang random dengan angka random. Dimensi yang dipilih dapat bervariasi dari hanya satu dimensi saja hingga keseluruhan dimensi. Organisme X_j adalah organisme selain X_i yang dipilih secara random dari ekosistem dan memiliki peran serupa tubuh manusia yang dijadikan inang oleh parasit plasmodium. Organisme X_j terserang oleh Parasite_Vector dan Parasite_Vector dapat menggantikan keberadaan organisme X_j di dalam ekosistem. Apabila Parasite_Vector lebih baik daripada organisme X_j, maka Parasite_Vector akan membunuh organisme X_j dan diterima sebagai organisme pengganti di dalam ekosistem. Sedangkan organisme X_j akan bertahan jika X_j lebih baik daripada Parasite_Vector karena X_j memiliki imunitas pertahanan diri yang cukup dan Parasite_Vector tidak dapat bertahan hidup.

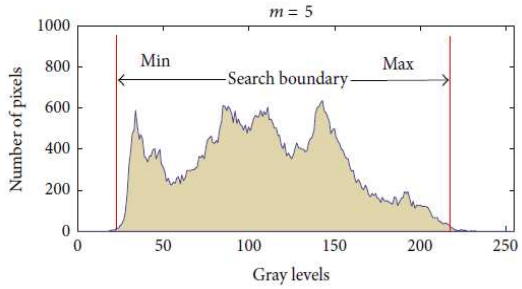
III. METODE DAN INTI PENELITIAN

Permasalahan multilevel image thresholding berurusan dengan mencari nilai beberapa threshold yang memaksimalkan J(t). Dimensi (m) pencarian untuk masalah optimisasi ditentukan berdasarkan banyaknya threshold. Gambar 2 merupakan flowchart bagaimana penelitian ini bekerja.



Gambar. 2. Flowchart Alur Penelitian

FA [7,8], SOS, dan IBA [9] digunakan untuk mencari kumpulan nilai threshold. Pada metode optimisasi dengan algoritma metaheuristic, adanya search boundary dapat mempercepat program dalam menemukan nilai optimal. Ketika banyaknya dimensi (m) ditingkatkan, maka kerumitan untuk masalah optimisasi meningkat pula. Penggunaan search boundary terdapat pada Gambar 3.



Gambar. 3. Histogram Gray Level

Gambar 3 menunjukkan histogram gray-level menggunakan search boundary. Area pencarian dengan search boundary menjadi [20, 220] daripada area pencarian awal [0, 255]. Adanya search boundary memungkinkan algoritma menemukan nilai optimal pada iterasi yang lebih kecil.

IV. UJI COBA

Percobaan dilakukan dengan berbagai macam gambar grayscale. Semua ukuran gambar yang dipakai disesuaikan menjadi 256x256 pixel agar dapat dibuat perbandingannya dengan seimbang, terutama dalam perhitungan waktu. Percobaan multilevel image thresholding menggunakan metode Otsu untuk mencari nilai optimal secara manual atau tanpa menggunakan algoritma metaheuristic tidak dilakukan karena beberapa penelitian telah mencobanya dan waktu perhitungan akan meningkat secara eksponensial sesuai dengan banyaknya dimensi (m), sehingga setiap algoritma akan selesai ketika iterasi maksimal.

Besarnya dimensi (m) yang dipakai adalah 2, 3, 4, dan 5. Karena algoritma multilevel image thresholding memiliki pengaruh nilai random, setiap percobaan diulang sebanyak 10 kali untuk gambar dan parameter yang sama. Uji coba dilakukan terlebih dahulu dengan membandingkan parameter besarnya populasi. Setelah itu, besarnya iterasi juga dibandingkan untuk mengetahui batasan iterasi ideal yang digunakan. Pada tahap akhir, uji coba menggunakan kombinasi besarnya populasi dan iterasi yang terbaik untuk menguji 20 gambar grayscale berbeda. Semua uji coba menggunakan Microsoft Visual Studio dengan bahasa C#.

Uji Coba 1 (parameter populasi). Pada tiga algoritma yang telah dibandingkan, banyak populasi (Po) ditentukan terlebih dahulu sebelum menentukan banyak iterasi (Itr) dengan uji coba perbandingan parameter populasi. Pada uji coba ini, banyak iterasi ditetapkan bernilai 10 dengan dimensi m bernilai 5 (dimensi tertinggi untuk uji coba) dan hanya menggunakan satu gambar saja (*Barbara*) untuk melihat bagaimana pengaruh perubahan parameter populasi terhadap hasilnya. Uji coba ini diawali dengan parameter populasi sebesar 3 untuk masing-masing algoritma metaheuristic yang digunakan dan terus ditingkatkan agar

dapat memantau pengaruhnya.

TABEL I
PERFORMA FIREFLY ALGORITHM (POPULASI)

Po	Itr	Best	Mean	Std	Std / Mean (%)	Avg Time (ms)
3	10	2554.59	2150.98	376.05	17.48	12.80
5	10	2654.63	2382.36	259.41	10.89	13.20
10	10	2668.18	2481.18	168.18	6.78	29.20
15	10	2693.00	2504.42	155.80	6.22	52.80
20	10	2681.24	2594.20	68.48	2.64	86.70
40	10	2651.69	2587.09	63.07	2.44	310.20
60	10	2678.53	2597.45	81.00	3.12	675.00

TABEL II
PERFORMA SYMBIOTIC ORGANISMS SEARCH (POPULASI)

Po	Itr	Best	Mean	Std	Std / Mean (%)	Avg Time (ms)
3	10	2687.59	2641.26	39.62	1.50	13.30
5	10	2700.91	2645.35	73.93	2.79	16.00
10	10	2699.55	2691.34	10.12	0.38	22.60
15	10	2700.75	2691.56	13.95	0.52	31.10
20	10	2702.41	2699.93	3.18	0.12	39.30
40	10	2702.43	2699.66	4.63	0.17	71.30
60	10	2702.38	2701.92	0.74	0.03	102.80

TABEL III
PERFORMA IMPROVED BAT ALGORITHM (POPULASI)

Po	Itr	Best	Mean	Std	Std / Mean (%)	Avg Time (ms)
3	10	2696.27	2641.28	69.39	2.63	14.60
5	10	2700.16	2690.97	9.17	0.34	15.80
10	10	2701.18	2697.99	3.58	0.13	20.00
15	10	2702.46	2699.80	2.10	0.08	27.30
20	10	2702.34	2698.32	3.42	0.13	32.00
40	10	2702.41	2699.85	2.49	0.09	54.60
60	10	2701.90	2699.39	3.40	0.13	84.20

Tabel 1, 2, dan 3 mencantumkan solusi terbaik (Best) untuk beberapa macam nilai populasi. Akan tetapi, dibutuhkan perhitungan nilai rata-rata (Mean) dan standar deviasi atau besarnya penyimpangan (Std) untuk menentukan kestabilan algoritma yang digunakan. Pada uji coba ini, populasi bernilai 20 dianggap cukup stabil untuk digunakan.

Uji Coba 2 (parameter iterasi). Pada tiga algoritma yang telah dibandingkan, banyak populasi ditentukan bernilai 20 untuk uji coba perbandingan parameter iterasi. Pada uji coba ini, dimensi m bernilai 5 (dimensi tertinggi untuk uji coba) dan hanya menggunakan satu gambar saja (Barbara) untuk melihat bagaimana pengaruh perubahan parameter iterasi terhadap hasilnya. Uji coba ini diawali dengan parameter iterasi sebesar 3 untuk masing-masing algoritma metaheuristic yang digunakan dan terus ditingkatkan agar dapat memantau pengaruhnya.

TABEL IV
PERFORMA FIREFLY ALGORITHM (ITERASI)

Po	Itr	Best	Mean	Std	Std / Mean (%)	Avg Time (ms)
20	**3**	2633.73	2516.19	116.4	4.63	26.60
20	**5**	2675.34	2549.90	94.74	3.72	43.60
20	**10**	2639.93	2594.70	45.81	1.77	90.10
20	**20**	2666.49	2576.26	91.15	3.54	208.70
20	**40**	2669.85	2588.24	57.72	2.23	417.50
20	**60**	2658.11	2612.18	33.75	1.29	498.20
20	**80**	2663.68	2628.17	32.48	1.24	837.10
20	**100**	2676.62	2637.25	31.57	1.20	1030.40
20	**120**	2690.76	2659.34	22.67	0.85	1262.90
20	**200**	2699.00	2689.96	6.10	0.23	2018.40

TABEL V
PERFORMA SYMBIOTIC ORGANISMS SEARCH (ITERASI)

Po	Itr	Best	Mean	Std	Std / Mean (%)	Avg Time (ms)
20	**3**	2696.21	2673.92	12.98	0.49	12.60
20	**5**	2699.64	2687.35	8.93	0.33	26.40
20	**10**	2701.80	2697.74	8.93	0.33	49.80
20	**20**	2702.60	2699.97	3.44	0.13	100.10
20	**40**	2702.63	2701.16	2.51	0.09	203.10
20	**60**	2702.63	2701.19	3.43	0.13	291.30
20	**80**	2702.63	2699.94	6.89	0.26	405.80

TABEL VI
PERFORMA IMPROVED BAT ALGORITHM (ITERASI)

Po	Itr	Best	Mean	Std	Std / Mean (%)	Avg Time (ms)
20	**3**	2696.28	2678.11	22.00	0.82	8.70
20	**5**	2700.31	2691.79	11.04	0.41	15.00
20	**10**	2702.20	2699.02	2.59	0.10	29.40
20	**20**	2702.45	2701.45	1.15	0.04	57.20
20	**40**	2702.63	2702.02	1.09	0.04	124.80
20	**60**	2702.60	2702.09	0.73	0.03	175.90
20	**80**	2702.63	2702.56	0.10	0.00	234.00

Tabel 4, 5, dan 6 mencantumkan solusi terbaik (Best) beserta kestabilan untuk beberapa macam nilai iterasi. Nilai iterasi sebesar 60 ditentukan cukup stabil pada kasus ini dan akan digunakan untuk tahapan uji coba berikutnya.

Uji Coba 3 (berbagai gambar dan dimensi). Uji coba bagian ini menggunakan 20 gambar grayscale berbeda. Semua gambar yang digunakan memiliki histogram gray-level yang bersifat unik. Sebagian besar gambar dianggap sulit untuk disegmentasi karena memiliki histogram yang bersifat multimodal atau memiliki banyak local optima. Untuk uji coba ini, setiap gambar diujikan dengan bermacam dimensi (m). Dimensi yang digunakan bernilai antara 2 hingga 5 dan nilai dimensi tersebut menandakan banyaknya threshold yang digunakan untuk memproses

gambar grayscale. Parameter populasi yang digunakan bernilai 20 dan parameter iterasi yang digunakan bernilai 60. Uji coba dilakukan sebanyak 10 kali untuk parameter dan gambar yang sama. Tabel 7 menunjukkan hasil uji coba menggunakan 7 dari keseluruhan 20 gambar uji coba (warna hijau menandakan hasil terbaik, kuning menandakan hasil terbaik kedua, dan merah menandakan hasil terburuk).

TABEL VII
PERBANDINGAN PERFORMA ALGORITMA

Pic	m	Std/Mean (%)			Average Time (ms)		
		FA	SOS	IBA	FA	SOS	IBA
Aerial	2	2.80	0.00	0.03	472.7	215.3	130.1
	3	2.03	0.00	0.30	478.8	209.0	149.9
	4	2.10	0.03	0.36	481.7	218.3	177.0
	5	2.00	0.14	0.13	496.2	224.4	204.2
Barbara	2	1.52	0.00	0.03	468.4	202.8	147.0
	3	2.71	0.00	0.04	485.5	205.9	161.3
	4	1.75	0.03	0.02	493.4	213.5	172.2
	5	1.92	0.17	0.02	498.2	219.7	193.8
Boats	2	1.00	0.00	0.03	474.5	198.1	139.4
	3	1.82	0.00	0.06	480.5	206.3	157.8
	4	1.32	0.04	0.06	499.5	213.6	173.2
	5	1.08	0.07	0.02	498.3	221.2	189.5
Goldhill	2	5.16	0.00	0.09	466.8	202.9	144.2
	3	3.94	0.00	0.06	480.8	209.0	164.3
	4	2.10	0.00	0.08	491.9	229.1	177.3
	5	2.15	0.15	0.10	502.9	224.6	190.1
Jet	2	0.96	0.00	0.28	460.5	201.3	148.0
	3	0.78	0.00	0.25	488.7	209.0	162.0
	4	1.48	0.01	0.30	496.7	215.1	185.6
	5	1.91	0.09	0.06	502.8	225.9	195.6
Lake	2	0.44	0.00	0.07	478.9	209.0	144.4
	3	1.43	0.00	0.01	487.2	212.8	169.9
	4	1.78	0.05	0.01	502.8	221.5	185.7
	5	1.41	0.01	0.03	493.4	224.4	195.9
Mandrill	2	2.23	0.00	0.01	464.5	194.9	146.2
	3	3.66	0.00	0.23	501.2	209.0	155.4
	4	1.49	0.08	0.10	496.6	210.5	157.4
	5	1.99	0.06	0.07	493.4	218.1	172.4

Berikut adalah salah satu contoh hasil threshold dan gambar hasil (gambar 4) dari tabel 7 untuk gambar Barbara dengan algoritma SOS:

1. $m = 2$; $J(t) = 2427.38$; $t = [81.69, 144.69]$
2. $m = 3$; $J(t) = 2594.87$; $t = [76.44, 127.8, 175.13]$
3. $m = 4$; $J(t) = 2666.67$; $t = [66.87, 106.76, 141.59, 181.43]$
4. $m = 5$; $J(t) = 2702.63$; $t = [56.29, 87.28, 117.14, 146.5, 182.8]$

Gambar. 4. Hasil Threshold Gambar Barbara.
(A) – Versi Greylevel
(B) – Versi Berwarna

V. KESIMPULAN

Kesimpulan yang diperoleh melalui penelitian ini adalah sebagai berikut:

1. Pertambahan populasi menyebabkan bertambahnya waktu perhitungan pada FA secara lebih signifikan daripada pertambahan waktu pada SOS dan IBA.
2. Pertambahan nilai dimensi cenderung menyebabkan bertambahnya waktu perhitungan. Pertambahan waktu pada FA dan SOS dapat dikatakan kurang signifikan apabila dibandingkan dengan pertambahan waktu pada IBA, karena IBA memiliki modifikasi operator crossover dan mutation pada setiap dimensinya.
3. Nilai terbaik yang didapatkan dengan SOS dan IBA lebih optimal daripada nilai terbaik yang didapatkan dengan FA.
4. Kestabilan SOS dan IBA lebih baik daripada FA secara signifikan, walaupun SOS memiliki keunggulan terhadap IBA pada dimensi yang kecil.
5. Waktu rata-rata perhitungan IBA paling cepat dan FA paling lama. Bila diurutkan waktunya, IBA < SOS < FA.

DAFTAR PUSTAKA

[1] Nobuyuki Otsu. 1979. *A Thresshold Selection Method from Graylevel Histogram*. IEEE Transaction on Systems, Man, and Cybernetics, vol. SMC-27, no. 1.

[2] Jun Zhang, Jinglu Hu. 2009. *Renal Biopsy Image Segmentation Based on 2-D Otsu Method wth Histogram Analysis*. Medcal Imaging Technology, vol. 27, no. 3.
[3] Ping-Sung Liao, Tse-Sheng Chen, Pau-Choo Chung. 2001. *A Fast Algorithm for Multilevel Thresholding*. Journal of Information Science and Engineering 17.
[4] X.-S. Yang. 2009. *Firefly Algorithms for Multimodal Optimization in Stochastic Algorithms: Foundations and Applizations*, vol. 5792 of *Lecture Notes in Computer Science*, pp. 167-178, Springer, Berlin, Germany.
[5] X.-S. Yang. 2010. *A New Metaheuristic Bat-inspired Algorithm, Studies in Computational Intelligence*, vol. 284, pp. 65-74, Springer, Berlin, Germany.
[6] Min-Yuan Cheng, Doddy Prayogo. 2014. *Symbiotic Organisms Search: A newmetaheuristic optimization algorithm*. Science Direct.
[7] N. Sri Madhava Raja, V. Rajinikanth, K. Lantha. 2014. *Otsu Based Optimal Multilevel Image THresholding uing Firefly Algorithm*. Hindawi Publishing Corporation.
[8] Kai Chen, Yifan Zhou, Zhisheng Zhang, Min Dai, Yuan Chao, Jinfei Shi. 2016. *Multilevel Image Segmentation Based on an Improved Firefly Algorithm*. Hindawi Publishing Corporation.
[9] Adis Alihodzic, Milan Tuba. 2014. *Improved Bat Algorithm Applied to Multilevel Image Threshodling*. Hindawi Publishing Corporation.

C. Pickerling lahir di Surabaya, Indonesia, pada tahun 1986. Menyelesaikan studi S1 di program studi Teknik Informatika STTS pada tahun 2008. Menyelesaikan studi masternya pada program studi Teknologi Informasi STTS pada tahun 2016. Minat penelitian adalah pada bidang ilmu software engineering dan evolutionary algorithm.

Hendrawan Armanto lahir di Surabaya, Indonesia, pada tahun 1986. Menyelesaikan studi S1 di program studi Teknik Informatika STTS pada tahun 2008. Menyelesaikan studi masternya pada program studi Teknologi Informasi STTS. Minat penelitian adalah bidang Artificial Intelligent, Evolutionary Algorithm, dan Game Development.