Perhitungan Objek Pada Citra Menggunakan Teknik *Thresholding*

Sharon Bernadetha Marbun / 13519092
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13519092@std.stei.itb.ac.id

***Abstract*—Perhitungan objek pada citra merupakan salah satu pekerjaan yang membutuhkan otomasi terutama apabila perhitungan objek dilakukan dalam jumlah yang besar. Terdapat berbagai metode untuk melakukan perhitungan objek secara otomatis pada citra, mulai dari metode yang berbasis pemrosesan citra hingga memanfaatkan intelegensi buatan dan pembelajaran mesin. Di dalam makalah ini, dibahas mengenai pemanfaatan pemrosesan citra dalam melakukan perhitungan objek pada citra, yaitu dengan menggunakan teknik *thresholding*. Pembuatan program perhitungan objek pada citra diimplementasikan dengan menggunakan kakas MATLAB.**

Keywords—perhitungan objek, pengolahan citra, teknik thresholding, MATLAB

I. PENDAHULUAN

Dewasa ini, pengolahan citra merupakan teknologi yang berkembang pesat, dan dianggap sebagai salah satu area penelitian yang penting di dalam ilmu komputer. Pengolahan citra adalah proses melakukan operasi pada citra, untuk meningkatkan kualitas citra atau untuk mengekstraksi informasi yang berguna dari citra tersebut. Perkembangan yang lebih lanjut menunjukkan bahwa tidak hanya pengolahan citra saja, citra juga dapat melalui tahapan analisis lebih lanjut menggunakan intelegensi buatan dan pembelajaran mendalam, yang dibahas di dalam area keilmuan *computer vision*.

Pemrosesan citra banyak digunakan untuk membantu pekerjaan manusia sehari-hari. Salah satu bentuk aplikasinya yaitu untuk melakukan perhitungan objek, yang dibahas secara khusus di dalam makalah ini. Meskipun dapat dengan mudah dilakukan, perhitungan objek secara manual dalam jumlah yang besar tentunya boros waktu dan tenaga. Apalagi, pekerjaan yang sifatnya repetitif ini menyia-nyiakan sumber daya manusia yang seharusnya bisa dimanfaatkan untuk pekerjaan lain yang membutuhkan kemampuan berpikir yang lebih tinggi. Maka dari itu, banyak penelitian yang dilakukan untuk dapat mengotomasi perhitungan objek baik pada gambar maupun video.

Perhitungan objek muncul dalam berbagai domain, seperti agrikultur (menghitung jumlah pohon), ekonomi dan pembangunan (menghitung jumlah bangunan untuk mengevaluasi populasi), biodiversitas (menghitung populasi), industri (menghitung jumlah produksi), dan masih banyak lagi.

Gambar 1.1 Contoh aplikasi perhitungan objek

Mengetahui hal ini, penulis merancang sebuah solusi untuk melakukan perhitungan objek sederhana dengan menggunakan salah satu teknik pemrosesan citra, yaitu *thresholding*.

II. LANDASAN TEORI

A. *Segmentasi Citra*

Segmentasi citra adalah operasi mempartisi citra menjadi sebuah koleksi yang terdiri dari sekumpulan pixel yang terhubung satu salama lain. Koleksi tersebut dapat berupa:

- region-region, yang biasanya mencakup keseluruhan citra
- struktur linier, seperti segmen garis dan segmen kurva
- bentuk-bentuk 2D, seperti lingkaran, elips, persegi, dan lain-lain.

Segmentasi citra menjadi sejumlah region bertujuan untuk membagi citra menjadi segmen-segmen atau objek-objek yang berbeda, ataupun untuk memisahkan objek dengan latar belakangnya. Dengan membagi citra menjadi sejumlah segmen, kita dapat memproses hanya segmen tertentu saja di dalam citra daripada memproses seluruh bagian citra. Segmen seperti ini disebut sebagai ROI (*region of interest*).

Tujuan dari segmentasi citra adalah menemukan bagian citra yang koheren atau objek spesifik. Citra disegmentasi berdasarkan properti yang dipilih seperti kecerahan, warna, tekstur, dan sebagainya. Segmentasi membagi citra menjadi sejumlah region yang terhubung. Tiap region bersifat homogen berdasarkan properti yang dipilih. Segmentasi citra merupakan

tahapn sebelum melakukan *image/object recognition*, *image understanding*, dan lain-lain.

Terdapat beberapa metode segmentasi citra yang umumnya dikelompokkan berdassarkan dua pendekatan sebagai berikut.

1. Diskontinuitas

   Metode ini mempartisi citra berdasarkan perubahan nilai intensitas *pixel* yang cepat, misalnya pendeteksian tepi.

2. *Similarity*

   Metode ini mempartisi citra berdasarkan kemiripan area menurut properti yang ditentukan. Metode segmentasi citra yang termasuk ke dalam pendekatan ini antara lain *thresholding*, *region growing*, *split and merge*, dan *clustering*.

B. *Thresholding*

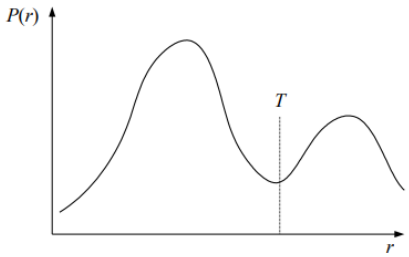
Dalam pemrosesan citra digital, teknik *thresholding*/ pengambangan merupakan metode paling sederhana untuk melakukan segmentasi citra. *Thresholding* dapat digunakan untuk mengubah citra *grayscale* menjadi citra biner.

Operasi pengambangan mengelompokkan nilai derajat keabuan setiap *pixel* ke dalam dua kelas, yaitu hitam dan putih. Formula umum operasi pengambangan adalah sebagai berikut.

$$f_B(i,j) = \begin{cases} 1, & f_g(i,j) \le T \\ 0, & \text{lainnya} \end{cases}$$

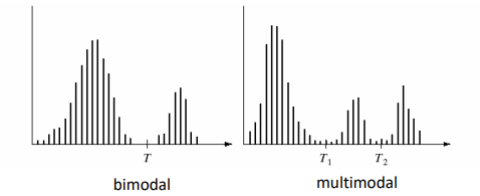
Berdasarkan rumus di atas, apabila nilai intensitas dari *pixel* adalah kurang dari atau sama dengan nilai ambang T, maka *pixel* tersebut diberi nilai 1. Sedangkan apabila nilai intensitas dari *pixel* melebihi nilai ambang T, *pixel* tersebut diberi nilai 0.

Untuk mendapatkan nilai ambang T, perlu dilakukan analisis histogram citra kemudian melakukan identifikasi puncah dan lembah. Nilai *grayscale* pada lembah terdalam di antara dua bukit dapat digunakan sebagai nilai T.



Gambar 2.1 Pemilihan nilai ambang T berdasarkan histogram citra

Namun, mencari nilai T dengan cara sederhana di atas hanya tepat jika histogram bersifat bimodal (mempunyai dua puncak dan satu lembah). Jika terdapat multimodal di dalam citra, diperlukan beberapa nilai ambang.

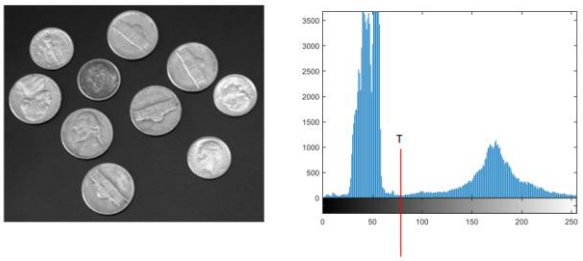


Gambar 2.2 Pemilihan nilai ambang pada citra bimodal dan multimodal

Terdapat tiga macam operasi pengambangan, yaitu sebagai berikut.

1. Pengambangan secara global

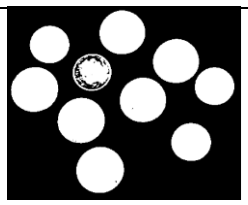
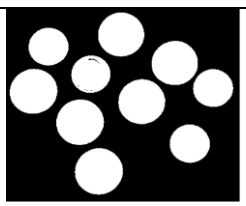
   Pengambangan secara global dilakukan dengan memilih sebuah nilai ambang T yang berlaku untuk seluruh bagian di dalam citra. Jika citra mengandung satu atau lebih objek dengan latar belakang dengan intensitas yang homogen, maka histogramnya akan bersifat bimodal (memiliki dua puncak).



Gambar 2.2 Pemilihan nilai ambang T pada citra koin

Dengan menganalisis histogram pada gambar 2.2, kita dapat memilih nilai ambang T yang paling optimal dalam melakukan segmentasi. Tabel berikut menunjukkan perbandingan hasil citra biner dengan menggunakan dua nilai T yang berbeda.

Tabel 2.1. Perbandingan citra biner hasil thresholding dengan nilai ambang T yang berbeda

Nilai T = 100	Nilai T = 75
	

Dengan menganalisis kedua hasil citra biner pada tabel 2.1, kita dapat menyimpulkan bahwa nilai ambang T = 75 menghasilkan citra biner yang lebih optimal dalam melakukan segmentasi objek.

2. Pengambangan secara lokal

   Pengambangan secara lokal dilakukan terhadap bagian-bagian tertentu di dalam citra. Dalam hal ini citra dipecah menjadi bagian-bagian kecil, kemudian proses pengambangan dilakukan secara lokal. Nilai ambang bergantung pada *pixel-pixel* bertetangga, sehingga nilai ambang untuk setiap bagian belum tentu sama dengan bagian lain.

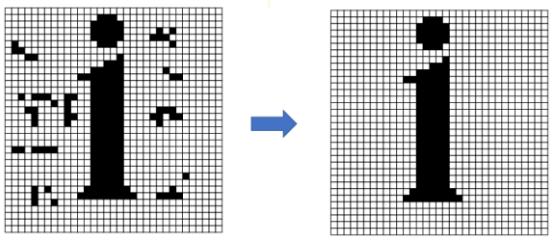
3. Pengambangan secara adaptif

   Pengambangan secara adaptif dapat mengubah nilai ambang secara dinamis pada citra. Metode pengambangan yang lebih canggih ini dapat mengakomodasi perubahan kondisi cahaya apda gambar, misalnya ketika terdapat gradian iluminasi yang kuat atau terdapat bayangan.

Citra biner hasil pengambangan dapat digunakan untuk memisahkan objek dengan latar belakang pada citra asalnya. Citra biner menjadi templat untuk melakukan segmentasi. Salah satu algoritma pengambangan yang sering digunakan untuk memisahkan objek dengan latar belakangnya adalah algoritma Otsu. Algoritma ini bekerja degna cara mengembalikan nilai ambang intensitas tunggal yang memisahkan *pixel* menjadi dua kelas, yaitu latar depan dan latar belakang. Nilai ambang ini ditentukan dengan meminimalkan variansi intensitas intra kelas, dan memaksimalkan variansi antar kelas.

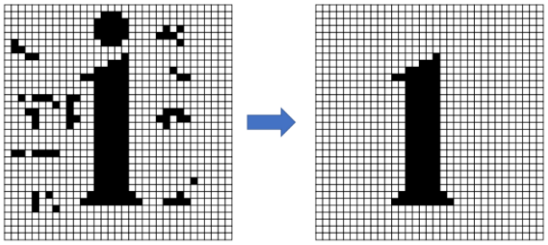
C. Penapis Luas

Citra biner hasil pengambangan terkadang mengandung beberapa daerah yang dianggap sebagai gangguan. Biasanya daerah gangguan tersebut berukuran kecil. Untuk menghilangkan daerah gangguan tersebut, dapat digunakan penapis luas. Misalkan objek yang dianalisis mempunyai luas yang lebih besar daripada T. Maka, *pixel-pixel* dari daerah yang luasnya di bawah T dinyatakan dengan 0 (dihilangkan).



Gambar 2.3 Hasil penapis luas dengan nilai T = 10

Gambar 2.3 menunjukkan transformasi citra biner setelah melalui penapis luas dengan nilai T = 10. Akan tetapi, perlu diperhatikan bahwa pemilihan nilai T yang tidak tepat dapat menyebabkan kesalahan, seperti pada gambar 2.4 berikut yang menunjukkan transformasi citra biner setelah melalui penapis luas dengan nilai T = 25.

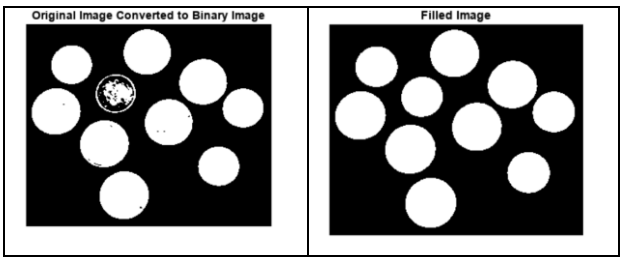


Gambar 2.3 Hasil penapis luas dengan nilai T = 25

III. METODOLOGI

Secara garis besar, langkah-langkah penyelesaian masalah yang penulis ajukan untuk melakukan perhitungan jumlah objek pada citra menggunakan teknik *thresholding* adalah sebagai berikut.

1. Pertama-tama, citra diubah dulu ke dalam bentuk *grayscale*.
2. Kemudian, citra *grayscale* tersebut diubah ke dalam bentuk biner dengan menggunakan algoritma *thresholding* yang dipilih. Citra biner yang dihasilkan menunjukkan hasil segmentasi antara objek-objek pada citra dengan latar belakangnya, sehingga hasilnya berupa objek berwarna putih dan latar belakangnya berwarna hitam.
3. Hasil segmentasi citra tersebut kemudian disempurnakan dengan mengisi bagian dalam objek yang "berlubang" (mengandung warna hitam) sehingga semua isinya menjadi berwarna putih. Contohnya ditunjukkan pada gambar 3.1 sebagai berikut.



Gambar 3.1 Hasil pengisian objek "berlubang"

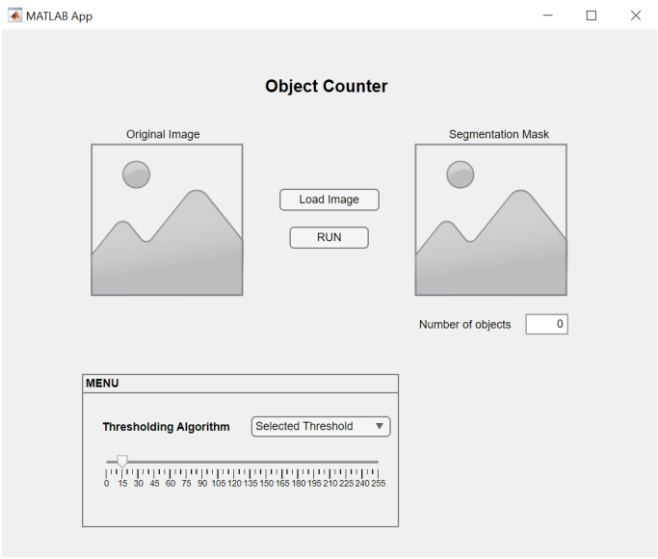
4. Kemudian, citra biner tersebut ditapis menggunakan penapis luas sehingga daerah dengan luas yang kecil (daerah gangguan) dapat dihapus.

IV. IMPLEMENTASI DAN HASIL

A. Implementasi

Solusi diimplementasikan menggunakan kakas MATLAB versi 2022a, dengan memanfaatkan *Image Processing Toolbox* dan MATLAB Designer App.

Tampilan awal GUI ditunjukkan pada gambar 4.1 sebagai berikut.



Gambar 4.1 Tampilan awal GUI

Pertama-tama, citra *input* dimuat dengan menekan tombol *Load Image* dan memilih citra *input* yang diinginkan pada *file explorer*. Citra *input* kemudian ditampilkan pada GUI.

Kemudian, pengguna memilih algoritma *thresholding* yang diinginkan. Terdapat tiga pilihan yang disediakan, yaitu menggunakan algoritma Otsu, menggunakan algoritma adaptif, atau pengguna memilih nilai ambang *threshold* yang diinginkan menggunakan *slider* (rentang nilai 0-255) yang disediakan. Setelah itu, dilakukan perhitungan objek pada citra sesuai dengan metode *thresholding* yang dipilih dengan menekan tombol *RUN*.

Dalam pemrosesannya, pertama-tama dilakukan pengubahan citra *input* ke dalam bentuk biner sesuai dengan algoritma *thresholding* yang dipilih.

```
% convert image to binary using thresholding method
function bw = getBinaryImg(img, threshold_algorithm, value)
    % convert to grayscale image
    gray = im2gray(img);

    % convert the grayscale image to binary image using selected method
    if (threshold_algorithm == "Selected Threshold")
        T = value/255;
        bw = imbinarize(gray, T);
    elseif (threshold_algorithm == "Otsu")
        T =  graythresh(gray);
        bw = imbinarize(gray, T);
    elseif (threshold_algorithm == "Adaptive")
        bw = imbinarize(gray,'adaptive');
    end
    % make sure that the background is black & the object is white
    if bwarea(bw) > bwarea(~bw)
        bw = ~bw;
    end
end
```

Kode program di atas menunjukkan fungsi *getBinaryImg* yang digunakan untuk mendapatkan citra biner yang diperoleh dari konversi citra *input*. Parameter yang digunakan adalah img, yaitu citra *input*, threshold_algorithm, yaitu *string* berupa metode *thresholding* yang dipilih, dan value, yaitu nilai ambang yang dipilih oleh pengguna apabila metode *thresholding* yang digunakan adalah "Selected Threshold" atau pemilihan nilai ambang secara manual. Untuk masing-masing metode *thresholding* sendiri diimplementasikan menggunakan fungsi *built-in* MATLAB yaitu *imbinarize*.

Citra biner yang diperoleh menunjukkan segmentasi antara objek-objek pada citra dengan latar belakangnya. Sebelum pemrosesan lebih lanjut untuk perhitungan jumlah objek pada citra, perlu dipastikan terlebih dahulu bahwa objek berwarna putih dan latar belakangnya berwarna hitam. Oleh karena itu dilakukan pengecekan apakah luas daerah berwarna putih lebih besar daripada luas daerah berwarna hitam. Apabila iya, maka dilakukan operasi komplemen terhadap citra biner tersebut, yaitu mengubah nilai *pixel* dari 1 menjadi 0 dan dari 0 menjadi 1 (putih menjadi hitam dan hitam menjadi putih).

Kemudian, citra biner tersebut melalui pemrosesan lebih lanjut untuk mendapatkan *segmentation mask*.

```
% get segmentation mask from binary image
function mask = getSegmentationMask(bw)
    % fill the holes in object
    mask = imfill(bw, 'holes');

    % remove objects with area less than 1500
    mask = bwareaopen(mask, 1500);
end
```

Kode program di atas menunjukkan fungsi *getSegmentationMask* untuk mendapatkan *segmentation mask* dari citra biner. Parameter yang digunakan adalah bw, yaitu citra biner hasil konversi dari citra *input*. Pertama-tama, dilakukan pengisian 'holes' pada citra dengan menggunakan fungsi *built-in* MATLAB yaitu *imfill*. Fungsi *imfill* bekerja dengan cara menemukan objek berwarna putih yang di dalamnya terdapat "lubang", yaitu *pixel* yang berwarna hitam. *Pixel* berwarna hitam yang dianggap sebagai lubang tersebut kemudian "diisi" sehingga diubah menjadi *pixel* berwarna putih.

Kemudian, untuk mengantisipasi kesalahan perhitungan jumlah objek, citra biner tersebut ditapis menggunakan penapis luas, sehingga objek dengan luas yang berada di bawah nilai ambang akan dihapus (diubah menjadi berwarna hitam). Hal ini dilakukan karena biasanya objek dengan luas yang kecil merupakan daerah gangguan (derau). Nilai ambang yang dipilih adalah 1500, sehingga daerah berwarna putih dengan

ukuran di bawah 1500 akan diubah menjadi berwarna hitam. Operasi ini dilakukan dengan menggunakan fungsi *built-in* MATLAB, yaitu *bwareaopen*. *Segmentation mask* kemudian ditampilkan pada GUI.

Kemudian, dari *segmentation mask* yang diperoleh, dilakukan perhitungan jumlah objek.

```
% get number of objects from the segmentation mask
function num = getNumberOfObjects(mask)
    num = bwconncomp(mask).NumObjects;
end
```

Kode program di atas menunjukkan fungsi *getNumberOfObjects* yang digunakan untuk mendapatkan jumlah objek yang dideteksi pada *segmentation mask*. Parameter yang digunakan adalah mask, yaitu *segmentation mask* yang diperoleh dari langkah sebelumnya. Perhitungan ini diimplementasikan menggunakan fungsi *built-in* MATLAB yaitu *bwconncomp*. Fungsi *bwconncomp* bekerja dengan cara mendeteksi dan menghitung jumlah komponen yang terhubung pada citra biner. Hasilnya adalah sebuah data berstruktur dengan komponen yaitu Connectivity, ImageSize, NumObjects, PixelIdxList. Contoh strukturnya ditunjukkan pada gambar 4.2 sebagai berikut.

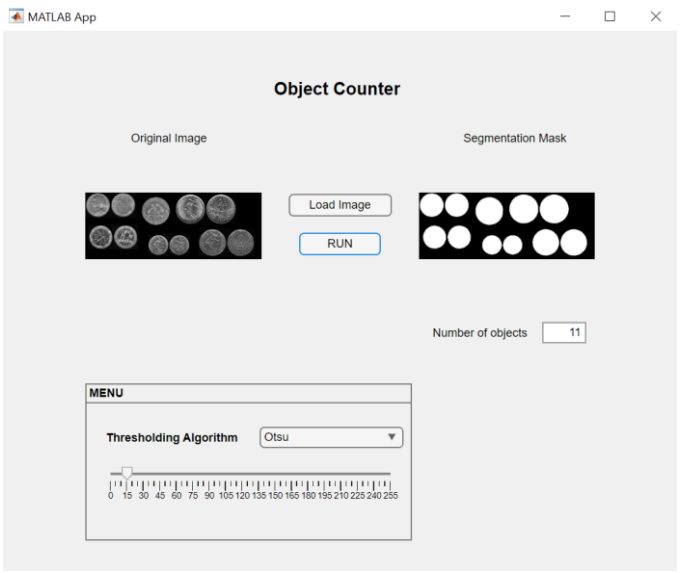
```
CC = bwconncomp(BW)

CC = struct with fields:
    Connectivity: 26
       ImageSize: [3 3 3]
      NumObjects: 2
    PixelIdxList: {[5x1 double]  [3x1 double]}
```

Gambar 4.2 Contoh struktur data hasil keluaran bwconnomp

Dilihat dari struktur di atas, maka nilai yang dikembalikan adalah komponen NumObjects, yang menunjukkan jumlah dari objek yang dideteksi dari citra biner *segmentation mask*. Hasil perhitungan ini kemudian ditampilkan pada GUI.

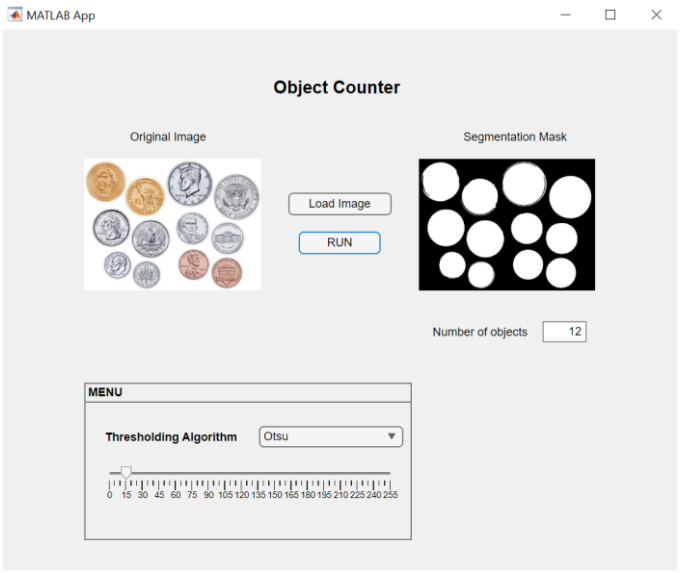
B. *Hasil dan Pembahasan*

Berikut merupakan hasil perhitungan objek dari beberapa citra uji. Hasil dari perhitungan objek pada citra uji yang ditunjukkan adalah hasil dari pemilihan metode *thresholding* yang terbaik untuk citra uji tersebut.



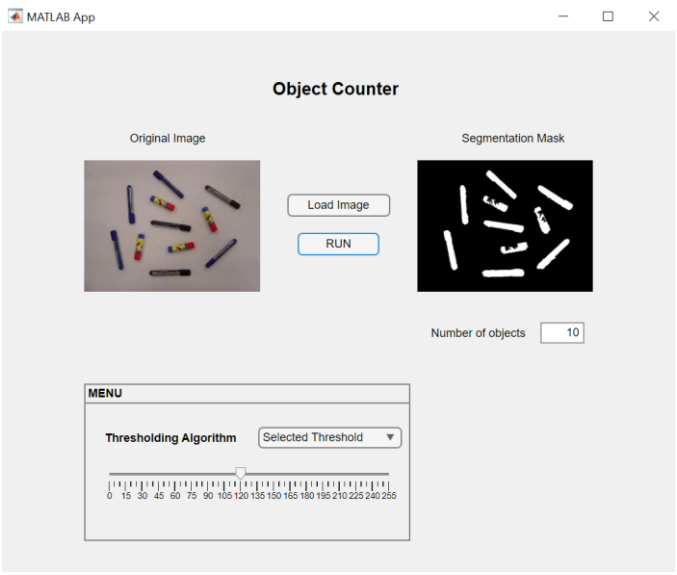
Gambar 4.3 Hasil citra uji 1 menggunakan metode thresholding Otsu

Dari hasil yang ditunjukkan pada gambar 4.3, dapat dilihat bahwa *segmentation mask* pada citra uji 1 terbentuk dengan baik, sehingga perhitungan objek menghasilkan akurasi sebesar 100%.



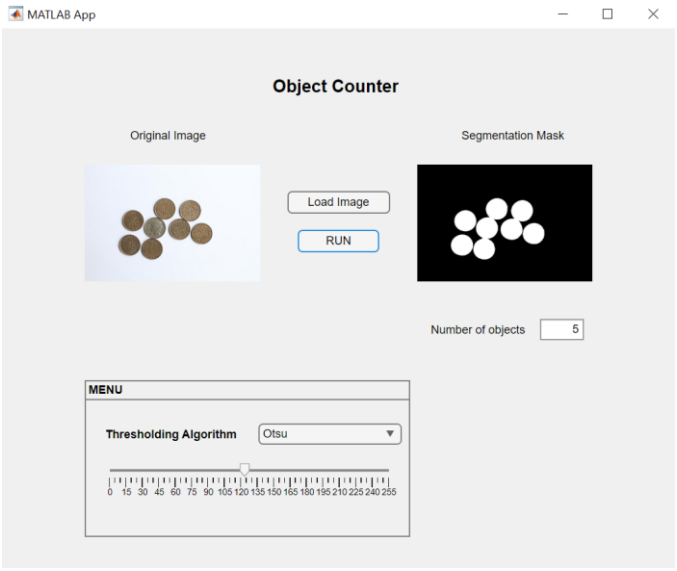
Gambar 4.4 Hasil citra uji 2 menggunakan metode thresholding Otsu

Dari hasil yang ditunjukkan pada gambar 4.4, dapat dilihat bahwa *segmentation mask* pada citra uji 2 terbentuk dengan baik, sehingga perhitungan objek menghasilkan akurasi sebesar 100%.



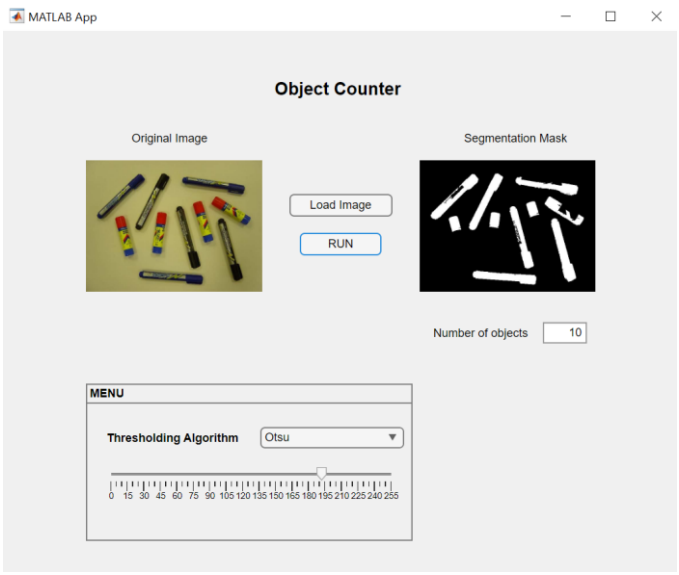
Gambar 4.5 Hasil citra uji 3 menggunakan threshold = 120

Dari hasil yang ditunjukkan pada gambar 4.5, dapat dilihat bahwa terdapat kesalahan pada pembuatan *segmentation mask*, yaitu satu buah objek tidak terdeteksi, sehingga perhitungan objek menghasilkan akurasi sebesar 90.91%. Hal ini dikarenakan intensitas antara objek dengan latar belakang tidak berbeda jauh, sehingga ketika melalui proses pengambangan, beberapa bagian dari objek disegmentasikan ke kelas yang sama dengan latar belakangnya.



Gambar 4.6 Hasil citra uji 4 menggunakan algoritma Otsu

Dari hasil yang ditunjukkan pada gambar 4.6, dapat dilihat bahwa segmentasi dapat dilakukan dengan baik. Namun, karena terdapat beberapa objek yang menempel, perhitungan objek tidak dapat dilakukan secara akurat karena algoritma yang diterapkan menghitung objek yang menempel sebagai satu kesatuan. Maka, untuk citra uji 4, akurasinya adalah 62.5%.

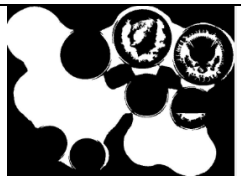


Gambar 4.7 Hasil citra uji 5 menggunakan algoritma Otsu

Dari hasil pada gambar 4.7, dapat dilihat bahwa *segmentation mask* yang dibentuk kurang baik karena tidak dapat menutupi seluruh bagian objek. Sama seperti citra uji 3, hal ini dikarenakan intensitas antara objek dengan latar belakangnya tidak berbeda jauh. Meskipun *segmentation mask* tidak terbentuk dengan sempurna, hasil perhitungan pada citra uji 5 memiliki akurasi sebesar 100%.

Dari pengujian yang dilakukan, penulis menemukan bahwa dibandingkan dengan algoritma Otsu, algoritma *thresholding adaptive* kurang optimal dalam menentukan nilai ambang yang cocok untuk melakukan segmentasi. Tabel berikut menunjukkan beberapa contoh *segmentation mask* yang dihasilkan menggunakan algoritma *adaptive*.

Tabel 4.1. Hasil segmentation mask menggunakan algoritma thresholding adaptive pada citra uji 2 dan 4

Citra *input*	*Segmentaion mask*
	
	

V. KESIMPULAN

Perhitungan objek dengan menggunakan teknik *thresholding* yang diimplementasikan di dalam pembuatan makalah ini dapat menghasilkan akurasi yang cukup baik, namun dengan batasan sebagai berikut.

- Latar belakang polos, sehingga ketika melalui proses pengambangan, latar belakang memiliki intensitas yang homogen sehingga disegmentasikan ke dalam satu kelas (*pixel* hitam).
- Terdapat perbedaan nilai intensitas yang cukup jauh antara objek dengan latar belakang, sehingga keduanya disegmentasikan ke dalam kelas yang berbeda (objek ke kelas putih dan latar belakang ke kelas hitam).
- Intensitas di antara objek-objek tidak jauh berbeda, sehingga ketika melalui proses pengambangan, objek disegmentasikan ke dalam kelas yang sama (*pixel* putih).
- Objek-objek tidak menempel, karena algoritma yang diterapkan melakukan perhitungan objek berdasarkan isolasi *pixel* berwarna putih (objek) oleh *pixel* hitam (latar belakang). Sehingga, objek yang menempel akan dihitung sebagai satu kesatuan.

Dari pengujian yang dilakukan, dapat disimpulkan bahwa algoritma Otsu dapat menghasilkan nilai ambang yang cukup baik dalam memisahkan objek dengan latar belakangnya, karena prinsip algoritmanya yang meminimalkan variansi intra-kelas dan memaksimalkan variansi antar kelas. Sedangkan untuk algoritma *adaptive* sendiri, tampaknya kurang cocok diimplementasikan dalam perhitungan objek ini, karena algoritma *adaptive* bekerja dengan cara menghitung nilai ambang berdasarkan region yang lebih kecil, sehingga nilai ambang bisa berbeda untuk setiap region. Sedangkan dalam perhitungan objek ini memfokuskan untuk memisahkan latar belakang dengan objek-objek yang umumnya homogen, sehingga nilai ambang lebih baik dipilih secara global.

PRANALA KODE PROGRAM

https://github.com/dethaa/ObjectCounting

UCAPAN TERIMA KASIH

Pertama-tama, penulis mengucapkan puji dan syukur kepada Tuhan Yang Maha Esa karena atas hikmat dan berkat yang diberikan-Nya, penulis dapat menyusun makalah ini hingga selesai. Penulis juga mengucapkan terima kasih kepada Bapak Dr. Rinaldi Munir, S.T., M.T. selaku dosen pengampu mata kuliah IF4073 Interpretasi dan Pengolahan Citra, atas ilmu yang disalurkan selama proses perkuliahan berlangsung. Penulis juga tidak lupa mengucapkan terima kasih kepada orang tua, saudara, dan rekan-rekan yang senantiasa mendukung dan memberikan motivasi kepada penulis.

REFERENSI

[1] E. Christophe and J. Inglada, "Object counting in high resolution remote sensing images with OTB," 2009 IEEE International Geoscience and Remote Sensing Symposium, 2009, pp. IV-737-IV-740, doi: 10.1109/IGARSS.2009.5417482.

[2] M. A. B. Siddique, R. B. Arif and M. M. R. Khan, "Digital Image Segmentation in Matlab: A Brief Study on OTSU's Image Thresholding," 2018 International Conference on Innovation in Engineering and Technology (ICIET), 2018, pp. 1-5, doi: 10.1109/CIET.2018.8660942.

[3] R. Munir, "*Citra Biner,"* 2022, https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2021-2022/20-Citra-Biner-2021.pdf

[4] R. Munir, "*Segmentasi Citra (Bagian 1),"* 2022, https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2021-2022/22-Segmentasi-Citra-Bagian1-2022.pdf

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Desember 2022

Sharon Bernadetha Marbun
13519092