Mendeteksi Blob dengan Menggunakan Algoritma BFS

Ahmad Fajar Prasetiyo (13514053)

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia

ahmad.fajar@students.itb.ac.id

***Abstract*— Saat ini teknologi berkembang dengan sangat pesat. Banyak sekali teknologi baru muncul. Teknologi yang sedang berkembang saat ini adalah teknologi robot. Dalam membuat robot *Artificial Intelligence* (AI) atau sering disebut kecerdasan buatan sangat berpengaruh penting. Salah satu bidang dalam kecerdasan buatan adalah *Computer Vision*. *Computer Vision* adalah cara bagaimana sebuah computer dapat mengolah suatu citra. Salah satu tahapan dalam *Computer Vision* adalah *Image Processing* atau sering disebut Pengolahan Citra. Salah satu teknik dalam pengolahan citra adalah menentukan *Blob* (kumpulan pixel). Blob ini dapat direpresentasikan dalam Graf static, sehingga dapat dideteksi dengan teknik penelusuran graf.**

Keywords-Kecerdasan buatan; computer vision; BFS; Color Filtering;

I. PENDAHULUAN

Saat ini teknologi berkembang dengan sangat pesat. Banyak sekali teknologi baru muncul. Teknologi yang sedang berkembang saat ini adalah teknologi robot. Contohnya sekarang China sudah membuat robot mirip wanita[1].

Dalam membuat suatu robot diperlukan berbagai macam teknik. Salah satu teknik yang paling menonjol adalah *Artificial Intelligence* (AI), atau sering disebut kecerdasan buatan.

Kecerdasan buatan adalah suatu sistem yang dapat berpikir dan bertindak menyerupai manusia[2]. Banyak bidang dalam kecerdasan buatan ini karena manusia memiliki banyak sekali faktor yang mempengaruhi cara manusia berfikir. Salah satu bidang dalam kecerdasan buatan adalah *Computer Vision*.

Computer Vision adalah suatu bidang yang mempelajari bagaimana cara membuat komputer dapat melihat seperti manusia melihat. Sehingga komputer bisa melihat objek secara prespektif 3D. Komputer juga bisa menentukan bahwa posisi benda. Selain itu dengan *Computer Vision* juga memungkinkan komputer untuk mengetahui kontur dari objek atau membedakan ini objek atau hanya latar belakang. Kita juga bisa membuat komputer bias membedakan ini bentuk persegi atau lingkaran. Bahkan lebih dari itu kita juga bisa membuat computer membedakan wajah atau menebak perasaan yang berdasarkan wajah[3]. *Computer Vision* memiliki banyak tingkatan yang menujukan tahap. Salah satu tahap awal dalam Computer Vision adalah *Image Proscessing* atau Pengolahan Citra.

Pengolahan Citra merupakan tahap paling awal dalam *Computer Vision*. Pengolahan Citra adalah tahap dimana citra diolah agar citra lebih mudah dalam diproses dan dianalisa pada tahap selanjutnya. Salah satu teknik dalam Pengolahan Citra adalah *Color Filtering*. Yang mana *Color Filtering* akan membentuk sebuah *Blob* atau sekumpulan pixel.

II. LANDASAN TEORI

A. *Traversal Graf*

Traversal Graf adalah suatu Algoritma yang digunakan

untuk mengunjungi simpul dalam Graf secara sistematis[4].

B. *Algoritma Pencarian Graf*

Algoritma Percarian Graf terbagi menjadi dua, yaitu tanpa informasi dan dengan informasi[4].

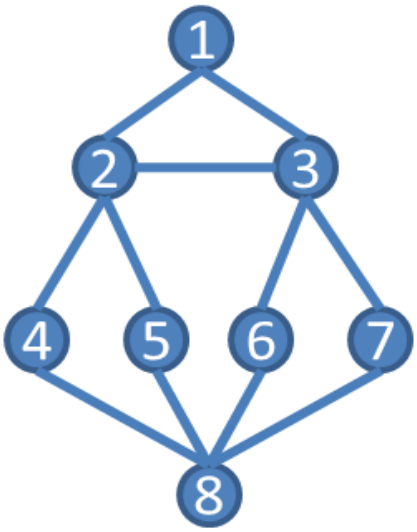
Teknik yang bisa digunakan dalam algoritma pencarian graf tanpa informasi adalah: *Depth First Search* (*DFS*), *Breath Fisrt Search* (*BFS*), *Depth Limited Search* (*DLS*), *Iterative Deepening Search*(*IDS*), *Uniform Cost Search*(*UCS*). Dalam pencarian menggunakan algoritma diatas tidak menggunakan informasi tambahan.

Teknik yang bisa digunakan dalam algoritma pencarian graf dengan informasi adalah: A* dan *Best Fisrt Search*. Pencarian dengan teknik ini biasanya menggunakan informasi *heuristic*. Biasanya teknik ini tau non-goal state.

C. *Representasi Graf dalam Proses Pencarian*

Dalam proses pencarian solusi graf dapat direpresentasikan dengan 2 macam yaitu:

- Graf statis: graf yang sudah terbentuk sebelum proses pencarian dilakukan.
- Graf dinamis: graf yang terbentuk saat proses pencarian dilakukan



Gambar 2.1 Graf Statis

D. *Pencarian Melebar (BFS) dalam Graf Statis*

Pencarian melebar pada graf statis.

Algoritma:

1. Kunjungi simpul v.
2. Kunjungi semua simpul yang bertetanggan dengan simpul v terlebih dahulu.
3. Kunjungi simpul yang belum dikunjungi dan bertentangga dengan simpul-simpul yang tadi dikunjungi, demikian seterusnya.

Struktur data:

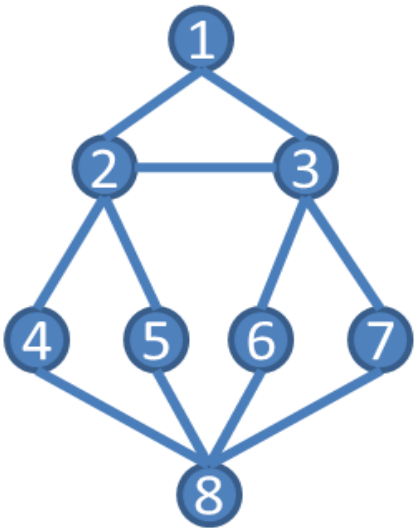
1. Matriks ketetanggaan A = [a_{ij}] yang berukuran nxn, a_{ij}= 1, jika simpul i dan simpul j bertetangga, a_{ij}= 0, jika simpul i dan simpul j tidak bertetangga.
2. Antrian q untuk menyimpan simpul yang telah dikunjungi.
3. Tabel Boolean, diberi nama "dikunjungi" **dikunjungi : array[l..n]** of boolean **dikunjungi[i]** = **true** jika simpul i sudah dikunjungi **dikunjungi[i]** = **false** jika simpul i belum dikunjungi.

Pseudo code:

```
procedure BFS(input v:integer)
{ Traversal graf dengan algoritma pencarian BFS.

  Masukan: v adalah simpul awal kunjungan
  Keluaran: semua simpul yang dikunjungi dicetak ke layar
}
Deklarasi
   w : integer
   q : antrian;

   procedure BuatAntrian(input/output q : antrian)
   { membuat antrian kosong, kepala(q) diisi 0 }

   procedure MasukAntrian(input/output q:antrian, input v:integer)
   { memasukkan v ke dalam antrian q pada posisi belakang }

   procedure HapusAntrian(input/output q:antrian,output v:integer)
   { menghapus v dari kepala antrian q }

   function AntrianKosong(input q:antrian) → boolean
   { true jika antrian q kosong, false jika sebaliknya }

Algoritma:
   BuatAntrian(q)          { buat antrian kosong }

   write(v)                { cetak simpul awal yang dikunjungi }
   dikunjungi[v]←true      { simpul v telah dikunjungi, tandai dengan
                             true}
   MasukAntrian(q,v)       { masukkan simpul awal kunjungan ke dalam
                             antrian}

  { kunjungi semua simpul graf selama antrian belum kosong }
   while not AntrianKosong(q) do
       HapusAntrian(q,v)   { simpul v telah dikunjungi, hapus dari
                             antrian }
       for tiap simpul w yang bertetangga dengan simpul v do
             if not dikunjungi[w] then
                write(w)     {cetak simpul yang dikunjungi}
                MasukAntrian(q,w)
                dikunjungi[w]←true
             endif
       endfor
   endwhile
  { AntrianKosong(q) }
```

Gambar 2.2 Pseudo Code BFS

Ilustrasi:



Iterasi	V	Q	dikunjungi							
			1	2	3	4	5	6	7	8
Inisialisasi	1	{1}	T	F	F	F	F	F	F	F
Iterasi 1	1	{2,3}	T	T	T	F	F	F	F	F
Iterasi 2	2	{3,4,5}	T	T	T	T	T	F	F	F
Iterasi 3	3	{4,5,6,7}	T	T	T	T	T	T	T	F
Iterasi 4	4	{5,6,7,8}	T	T	T	T	T	T	T	T
Iterasi 5	5	{6,7,8}	T	T	T	T	T	T	T	T
Iterasi 6	6	{7,8}	T	T	T	T	T	T	T	T
Iterasi 7	7	{8}	T	T	T	T	T	T	T	T
Iterasi 8	8	{}	T	T	T	T	T	T	T	T

Gambar 2.3 Ilustrasi BFS pada Graf Static

III. Eksperimen

Eksperimen ini ditulis dengan Bahasa C++ dengan menggunakan library OpenCV. OpenCV (Open Computer Vision) adalah sebuah API (Application Programming Interface) Library yang digunakan pada Pengolahan Citra Computer Vision.

Gambar yang akan digunakan dalam eksperimen kali ini adalah gambar Tupperware[6]. Gambar tersebut memiliki 2 warna yaitu warna hijau dan warna ungu. Karena tidak banyak terdapat variasi warna maka *Color Filtering* akan lebih mudah untuk dilakukan.

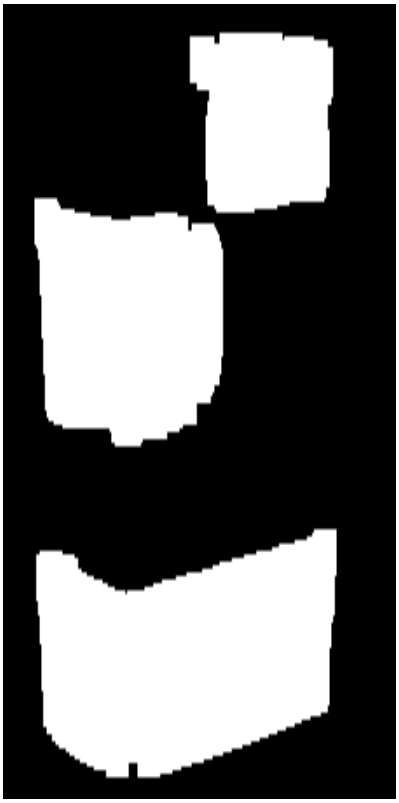
Color Filtering disini menggunakan suatu batas. Jika pada suatu pixel jarak nilai RGB pada pixel lebih besar dari batas yang kita tentukan dengan nilai RGB target pada kasus ini yaitu nilai RGB ungu. Jika nilainya lebih besar maka tidak masuk kedalam yang kita cari jadi kita warnai gambar dengan warna hitam yang menandakan dia tidak masuk filter. Sedangkan jika jarak RGB lebih kecil dari batas yang kita tentukan maka pixel tersebut masuk kedalam yang kita cari. Pixel yang masuk kedalam yang kita cari akan diwarnai putih.

Setelah *Color Filtering* ada tahap yang digunakan untuk menambal gambar yang tidak terdeteksi. Hal ini diperlukan karena pada setiap gambar terdapat gangguan (*noise*). Tahap ini tidak akan dibahas lebih lanjut karena konsen kita dalam masalah penelusuran graf bukan pada *Image Processing* atau Pengolahan Citra.

Gambar 3.1 Gambar masukan

Setelah gambar melewati tahap filtering gambar akan berubah hitam dan putih saja. Pixel yang berwarna putih adalah pixel yang masuk sementara pixel yang berwarna hitam adalah pixel yang tidak masuk. Dari sini dapat kita lihat bahwa pixel berwarna putih terkumpul atau yang sering kita sebut dengan blob. Setelah *color filtering* kita bisa langsung masuk ke tahap mendeteksi blob.

Cara mendeteksi blob ada beberapa cara, dalam eksperimen kali ini kita akan menggunakan *traversal* (penelusuran dari awal pixel sampai ahir) atau menggunakan algoritma penelusuran graf melebar (BFS).



Gambar 3.2 Gambar Threshold

A. Mendeteksi Blob dengan Traversal

Cara ini menggunakan penelusuran 1 per 1 pixel yang ada pada gambar. Pixel akan ditelusuri dan akan dicari nilai ordinat maksimal dan minimal, dan absis maksimal dan minimal. Setelah itu akan dibetuk suatu persegi panjang yang akan menunjukkan dimana blob itu.

Teknik ini memiliki banyak kelemahan. Karena dalam satu gambar kita tidak bisa memisahkan antar blob. Sehingga pixel yang tidak dicari pun bisa kena.

Teknik ini juga memiliki kelebihan karena tidak ada data yang harus disimpan maka teknik ini sangat cepat prosesnya. Tetapi dalam banyak kejadian teknik ini tidak bisa dilakukan.

Gambar 3.3 Gambar Traversal

Dapat dilihat bahwa hasilnya tidak sesuai dengan apa yang kita harapkan. Tetapi dalam kasus tertentu teknik ini bisa dilakukan. Teknik ini bisa dilakukan ketikan blob yang terbentuk hanya ada 1.

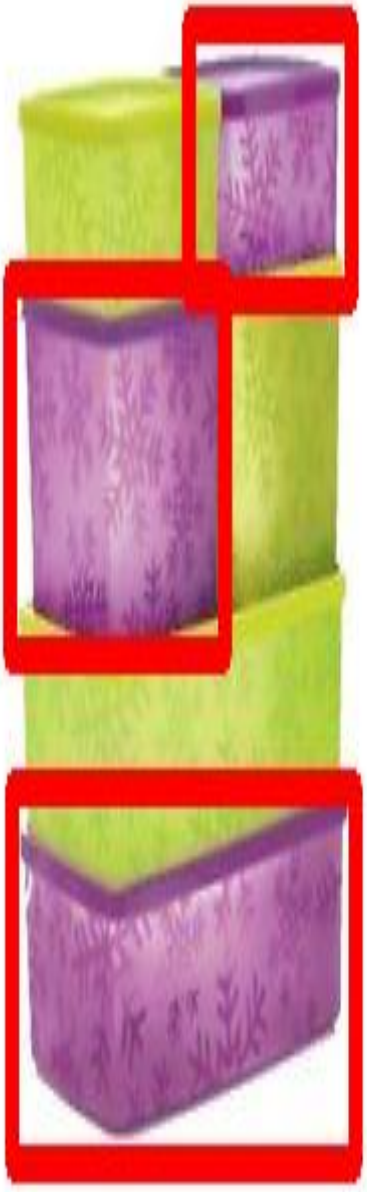
B. Mendeteksi Blob dengan BFS

Cara yang digunakan dalam mendeteksi blob ini mirip dengan yang traversal tetapi ada sedikit modifikasi. Awal kita definisikan suatu array of Boolean yang merepresentasikan bahwa pixel ini sudah dikunjungi atau tidak.

Setelah itu kita kunjungi mulai dari awal pixel. Dan setiap pixel yang telah kita kunjungi kita rubah aray of Boolean nya menjadi true. Jika kita mendapatkan hitam kita akan lanjutkan

ke dalam pixel yang selanjutnya. Jika kita bertemu dengan putih kita akan menerapkan algoritma BFS. Dengan simpul yang terkoneksi adalah putih sementari hitam adalah simpul yang tidak terhubung.

Dari setiap kita melakukan pencarian dengan BFS kita akan mencatat nilai ordinat maksimal dan minimal, dan nilai absis maksimal dan minimal. Dari nilai-nilai itu akan dibentuk suatu persegi panjang yang merepresentasikan bahwa persegi panjang itu adalah sebuah blob atau kumpulan pixel. Jadi setiap kali kita melakukan BFS kita akan mendapatkan suatu persegi panjang.



Gambar 3.4 Gambar BFS

Teknik ini memiliki kelebihan karena bisa mendeteksi lebih dari satu blob. Dan banyak kasus yang dapat diselesaikan dengan teknik ini. Dapat kita lihat bahwa blob yang terbentuk sama dengan pergi panjang yang dihasilkan.

Selain memiliki kelebihan teknik ini juga memiliki kelemahan. Waktu yang digunakan untuk menggunakan teknik ini terbilang cukup lama. Karena pada awal kita harus menginisialisasi sebuah array of Boolean. Selain itu setiap pemanggilan BFS kita harus juga menggunakan antrian (*queue*) yang mana akan terdapat suatu proses penyimpanan dan pengambilan suatu data. Kalau data yang diolah hanya satu bisa dapat ditangani, tetapi beda jika yang ditangani suatu aliran gambar atau video. Kita harus menggunakan computer yang lumayang bagus untuk mengolah setiap gambar tersebut. Dan mengatur delay pengambilan gambar agar tidak terjadi eror.

IV. KESIMPULAN

Mendekteksi blob bisa dilakukan banyak cara, dua contohnya adalah dengan menggunakan algoritma traversal dan menggunakan algoritma BFS.

Algoritma traversal hanya bisa menangani jika hanya terdapat satu blob dalam sebuah gambar. Sementara algoritma BFS dapat menangani jika terdapat lebih satu blob dalam sebuah gambar.

Waktu yang digunakan untuk menjalakan algoritma traversal jauh lebih singkat jika dibandingkan dengan menggunakan algoritma BFS. Hal ini dikarena algoritma traversal tidak perlu menginisialisai array of Boolean yang merepresentasikan bahwa pixel ini telah dikunjungi atau tidak sementara algoritma BFS perlu. Algoritma traversal tidak perlu untuk mencatat data pixel yang telah dikunjungi sementara algortima BFS perlu. Algortima traversal tidak perlu mengolah lagi pixel yang berwarna putih sementara algoritma BFS perlu hal ini digunakan untuk membangkitkan anak dari pixel tersebut.

Setiap algoritma memiliki kelemahan dan kelebihan. Tidak ada algoritma yang bagus dalam semua kondisi. Maka dari itu pemilihan algoritma sangat penting untuk menangani kondisi tertentu.

UCAPAN TERIMA KASIH

Pertama penulis ingin mengucapkan terima kasih kepada Tuhan yang Maha Esa atas hikmat dan waktu yang telah diberikan kepada penulis agar dapat menyelesaikan makalah ini. Tak lupa penulis juga mengucapkan terima kasih kepada kedua orang tua penulis karena tanpa jasa dan bimbingannya penulis tidak dapat menuntut ilmu di Intitut Teknologi Bandung dan menyelesaikan makalah ini. Penulis juga ingin mengucapkan terima kasih kepada Bapak Rinaldi Munir dan Ibu Ulfa karena melalui pengajarannya, saya dapat mengerti konsep Strategi Algoritma dan teori graf yang menjadi dasar makalah ini.

REFERENSI

[1] http://teknologi.news.viva.co.id/news/read/762052-jia-jia-robot-wanita-cantik-paling-mirip-manusia, diakses pada tanggal 08 Mei 2016.

[2] Russell , Stuart J. and Peter Norvig. Artificial Intelligence : A Modern Approach. New Jersey: Prentice-Hall, Inc. 1995.

[3] Szeliski, Richard. Computer Vision: Algorithms and Applications. New York: Springer. 2011.

[4] Slide kuliah IF2211 Strategi Algoritma 2016.

[5] http://www.priawadi.com/2012/09/opencv.html, diakses pada tanggal 08 Mei 2016.

[6] http://thumbs4.ebaystatic.com/d/l225/m/mdC5A5akVM8v46RsLx2GCZA.jpg, diakses pada tanggal 08 Mei 2016.

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, 9 Mei 2016

Ahmad Fajar Prasetiyo(13514053)