

## SKRIPSI

### PENERAPAN DATA MINING PADA MASALAH PENGENALAN POINT OF INTEREST



Gian Martin Dwibudi

NPM: 6181801015

PROGRAM STUDI TEKNIK INFORMATIKA  
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS  
UNIVERSITAS KATOLIK PARAHYANGAN

«tahun»



**UNDERGRADUATE THESIS**

**APPLICATION OF DATA MINING ON THE PROBLEM OF  
RECOGNIZING POINT OF INTEREST**



**Gian Martin Dwibudi**

**NPM: 6181801015**

**DEPARTMENT OF INFORMATICS  
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES  
PARAHYANGAN CATHOLIC UNIVERSITY**

**«tahun»**



## **LEMBAR PENGESAHAN**

### **PENERAPAN DATA MINING PADA MASALAH PENGENALAN POINT OF INTEREST**

**Gian Martin Dwibudi**

**NPM: 6181801015**

Bandung, «tanggal» «bulan» «tahun»

Menyetujui,

Pembimbing

/KDH

Ketua Tim Penguji

Anggota Tim Penguji

«penguji 1»

«penguji 2»

Mengetahui,

Ketua Program Studi

Mariskha Tri Adithia, P.D.Eng



## **PERNYATAAN**

Dengan ini saya yang bertandatangan di bawah ini menyatakan bahwa skripsi dengan judul:

### **PENERAPAN DATA MINING PADA MASALAH PENGENALAN POINT OF INTEREST**

adalah benar-benar karya saya sendiri, dan saya tidak melakukan penjiplakan atau pengutipan dengan cara-cara yang tidak sesuai dengan etika keilmuan yang berlaku dalam masyarakat keilmuan.

Atas pernyataan ini, saya siap menanggung segala risiko dan sanksi yang dijatuhan kepada saya, apabila di kemudian hari ditemukan adanya pelanggaran terhadap etika keilmuan dalam karya saya, atau jika ada tuntutan formal atau non-formal dari pihak lain berkaitan dengan keaslian karya saya ini.

Dinyatakan di Bandung,  
Tanggal «tanggal» «bulan» «tahun»



Gian Martin Dwibudi  
NPM: 6181801015



## **ABSTRAK**

«Tuliskan abstrak anda di sini, dalam bahasa Indonesia»

**Kata-kata kunci:** «Tuliskan di sini kata-kata kunci yang anda gunakan, dalam bahasa Indonesia»



## **ABSTRACT**

«Tuliskan abstrak anda di sini, dalam bahasa Inggris»

**Keywords:** «Tuliskan di sini kata-kata kunci yang anda gunakan, dalam bahasa Inggris»



*«kepada siapa anda mempersembahkan skripsi ini. . . ?»*



## **KATA PENGANTAR**

«Tuliskan kata pengantar dari anda di sini . . . »

Bandung, «bulan» «tahun»

Penulis



# DAFTAR ISI

<b>KATA PENGANTAR</b>	<b>xv</b>
<b>DAFTAR ISI</b>	<b>xvii</b>
<b>DAFTAR GAMBAR</b>	<b>xix</b>
<b>1 PENDAHULUAN</b>	<b>1</b>
1.1 Latar Belakang . . . . .	1
1.2 Rumusan Masalah . . . . .	2
1.3 Tujuan . . . . .	3
1.4 Batasan Masalah . . . . .	3
1.5 Metodologi . . . . .	3
1.6 Sistematika Pembahasan . . . . .	3
<b>2 LANDASAN TEORI</b>	<b>5</b>
2.1 Point of Interest . . . . .	5
2.2 Object Instance Recognition . . . . .	6
2.3 SIFT (Speeded Up Robust Feature) . . . . .	8
2.3.1 Pencarian Extrema . . . . .	8
2.3.2 Penentuan Skala . . . . .	10
2.3.3 Penentuan Orientasi . . . . .	11
2.3.4 Pembuatan Deskriptor . . . . .	13
2.4 ORB (Oriented FAST and Rotated BRIEF) . . . . .	13
2.4.1 Pencarian Keypoint . . . . .	13
2.4.2 Penentuan Skala . . . . .	14
2.4.3 Penentuan Orientasi . . . . .	14
2.4.4 Pembuatan Deskriptor . . . . .	15
2.5 BSIS (Best Score Increasing Subsequence) . . . . .	15
2.5.1 Pairing . . . . .	15
2.5.2 Verification . . . . .	15
2.5.3 Scoring . . . . .	16
2.6 KD-Tree . . . . .	16
2.7 Clustering . . . . .	16
2.7.1 Agglomerative . . . . .	16
2.7.2 DBSCAN . . . . .	16
<b>DAFTAR REFERENSI</b>	<b>19</b>
<b>A KODE PROGRAM</b>	<b>21</b>
<b>B HASIL EKSPERIMEN</b>	<b>23</b>



## DAFTAR GAMBAR

1.1	Contoh POI yang memiliki logo unik. POI seperti ini dapat dikenali dengan melakukkan identifikasi pada logo tersebut. . . . .	1
1.2	Empat gambar di atas menunjukkan permasalahan yang dihadapi pada penelitian ini. Gambar (a) dan (b) merupakan gambar dari dua POI yang berbeda tetapi memiliki logo dengan bagian sudut yang mirip, sudut yang mirip tersebut kemungkinan akan menghasilkan fitur lokal yang mirip juga. Sedangkan pada gambar (c) dan (d) merupakan dua gambar dari POI yang sama tetapi banyak objek latar yang berbeda sehingga akan memunculkan fitur lokal yang tidak konsisten. . . . .	2
2.1	Salah satu contoh POI . . . . .	5
2.2	Contoh POI dengan logo unik. . . . .	5
2.3	Ilustrasi fitur lokal yang konsisten dan tidak konsisten. Pasangan fitur lokal A1 dengan A2 konsisten secara geometris karena keduanya memiliki posisi yang sama relatif terhadap fitur lokal lain. Fitur lokal A1 berada di atas B1 dan C1 sedangkan fitur lokal A2 berada di atas B2 dan C2. Pasangan fitur lokal B1 dengan B2 dan C1 dengan C2 tidak konsisten secara geometris karena B1 berada di kanan C1 sedangkan B2 berada di sebelah kiri C2. . . . .	7
2.4	Kurva Gaussian dan bentuk representasi <i>matrix</i> -nya. . . . .	8
2.5	Kurva dan <i>matrix</i> Gaussian pada nilai $\sigma$ yang berbeda. . . . .	9
2.6	Efek nilai $\sigma$ pada hasil gambar konvolusi. . . . .	9
2.7	Operasi DoG pada gambar . . . . .	10
2.8	Penggunaan DoG pada SIFT . . . . .	10
2.9	Oktaf pada proses konvolusi SIFT . . . . .	11
2.10	Ilustrasi pembobotan pada <i>Gaussian Weighting</i> . Titik tengah merupakan <i>keypoint</i> yang diperiksa sedangkan setiap kotak merupakan <i>pixel-pixel</i> di sekitar <i>keypoint</i> . Tanda panah pada tiap kotak menunjukkan <i>magnitude</i> dan orientasi <i>pixel</i> tersebut, panjang panah merupakan nilai <i>magnitude</i> dan arahnya merupakan orientasi . . . . .	12
2.11	Histogram untuk menentukan orientasi dari <i>keypoint</i> . Bin dengan nilai tertinggi (tanda panah biru) akan digunakan sebagai orientasi dari <i>keypoint</i> . Untuk bin lain yang jumlahnya berada dalam rentang 80% dari bin tertinggi (tanda panah hijau) digunakan untuk membuat <i>keypoint baru</i> . . . . .	12
2.12	Ilustrasi <i>keypoint</i> pada ORB. Titik tengah pada gambar tersebut memiliki <i>pixel-pixel</i> dengan nilai jauh lebih kecil (lebih gelap) yang mengelilinginya. <i>Pixel-pixel</i> yang mengelilingi tersebut seakan membentuk sudut dengan titik tengah sebagai pusatnya. . . . .	14
2.13	<i>Image Pyramid</i> pada ORB . . . . .	14
B.1	Hasil 1 . . . . .	23
B.2	Hasil 2 . . . . .	23
B.3	Hasil 3 . . . . .	23
B.4	Hasil 4 . . . . .	23



# BAB 1

## PENDAHULUAN

### 1.1 Latar Belakang

Sebuah lokasi atau titik geografis yang memiliki kegunaan tertentu biasa disebut sebagai *Point of interest* (POI). POI ini dapat berupa tempat apa saja yang memiliki ciri khas tertentu dan dapat dikenali dari ciri khas tersebut. POI-POI tertentu dapat dikenali dari logo tempat tersebut atau objek-objek lainnya yang terlihat secara langsung. Seperti ditunjukkan pada Gambar 1.1, kedua POI tersebut memiliki logo unik yang terlihat dengan jelas.

Pada skripsi ini akan dibuat sebuah sistem untuk mengidentifikasi POI dari masukan yang berisi gambar POI tersebut. Proses identifikasi POI dilakukan dengan mendeteksi logo khusus atau objek unik yang ada pada POI tersebut.



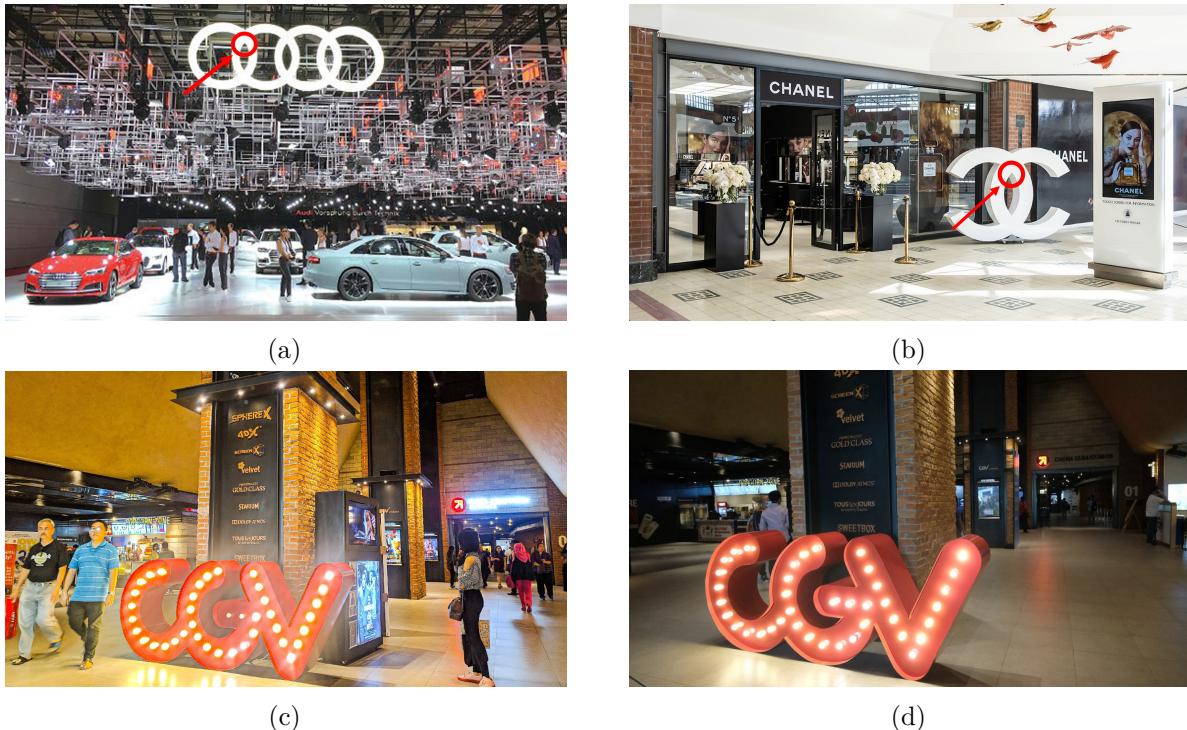
Gambar 1.1: Contoh POI yang memiliki logo unik. POI seperti ini dapat dikenali dengan melakukan identifikasi pada logo tersebut.

Proses identifikasi POI akan dilakukan menggunakan teknik *Object Instance Recognition* (OIR). Teknik OIR merupakan teknik pengenalan objek spesifik. Sebuah algoritma OIR harus dapat mengatasi masalah seperti pencahayaan, sudut pengambilan, objek *background*. Objek harus tetap dapat diidentifikasi walaupun gambar memiliki gangguan-gangguan tersebut. OIR dapat dilakukan dengan memanfaatkan fitur lokal.

Fitur lokal merupakan fitur yang mendeskripsikan sebuah daerah penting (*keypoint*) pada gambar. Salah satu cara mendapatkan *Keypoint* adalah dengan mencari sudut-sudut atau perpotongan garis yang terdapat pada gambar. *Keypoint-Keypoint* yang terdeteksi pada gambar akan memiliki sebuah vektor untuk mendeskripsikan daerah di sekitar *keypoint* tersebut yang disebut sebagai vektor deskriptor. Proses pencarian fitur lokal pada penelitian ini akan dilakukan dengan menggunakan metode *Scale Invariant Feature Transform* (SIFT) dan *Oriented FAST and Rotated BRIEF* (ORB). Metode SIFT dan ORB akan menghasilkan vektor deskriptor untuk tiap fitur lokal yang terdeteksi, vektor deskriptor ini dapat digunakan untuk mengidentifikasi fitur lokal.

Salah satu masalah yang ada pada pengenalan POI adalah pada sebuah gambar POI tidak semua fitur lokal yang dideteksi bersifat unik terhadap POI tersebut. Ada fitur lokal yang juga

dimiliki oleh POI lain atau fitur lokal yang berasal dari objek di latar belakang yang sifatnya tidak konsisten. Masalah ini akan mempersulit pada proses OIR untuk mengidentifikasi POI yang tepat. Gambar 1.2 menunjukkan masalah-masalah ini, gambar 1.2a dan 1.2b menunjukkan fitur lokal yang mirip dari dua POI yang berbeda, sedangkan gambar 1.2c dan 1.2d menunjukkan objek-objek latar belakang yang tidak konsisten pada POI. Penelitian ini akan melakukan analisis untuk menemukan dan memisahkan fitur-fitur lokal tersebut agar tidak diproses dalam pembuatan model POI.



Gambar 1.2: Empat gambar di atas menunjukkan permasalahan yang dihadapi pada penelitian ini. Gambar (a) dan (b) merupakan gambar dari dua POI yang berbeda tetapi memiliki logo dengan bagian sudut yang mirip, sudut yang mirip tersebut kemungkinan akan menghasilkan fitur lokal yang mirip juga. Sedangkan pada gambar (c) dan (d) merupakan dua gambar dari POI yang sama tetapi banyak objek latar yang berbeda sehingga akan memunculkan fitur lokal yang tidak konsisten.

Fitur-fitur lokal yang tidak unik dan tidak konsisten tersebut akan dipisahkan dengan menggunakan metode *clustering*. Metode *clustering* merupakan teknik pemrosesan data yang akan mengelompokkan data-data dengan sifat yang mirip ke dalam satu kelompok. Metode *clustering* pada penelitian ini akan menggunakan metode *Agglomerative* dan DBSCAN. Penelitian ini mengasumsikan fitur lokal yang merepresentasikan suatu POI adalah fitur lokal yang muncul secara konsisten di gambar POI tersebut dan relatif unik terhadap POI tersebut.

## 1.2 Rumusan Masalah

Skripsi ini memiliki rumusan masalah sebagai berikut:

- Bagaimana membuat model pengenalan POI berdasarkan fitur lokalnya menggunakan teknik *data mining*?
- Bagaimana mengidentifikasi POI dalam sebuah gambar berisi POI dengan memanfaatkan model pengenalan POI yang telah dibuat?

### 1.3 Tujuan

Skripsi ini memiliki tujuan sebagai berikut:

- Membuat perangkat lunak yang akan menghasilkan model pengenalan POI berdasarkan dari *dataset* yang diberikan
- Membuat perangkat lunak yang dapat melakukan identifikasi POI dari sebuah gambar POI dengan menggunakan model yang dihasilkan.

### 1.4 Batasan Masalah

Berikut batasan-batasan masalah dari skripsi ini:

- Pengambilan fitur lokal untuk analisis dilakukan menggunakan metode SIFT dan ORB dengan implementasi OpenCV pada Python.

### 1.5 Metodologi

Skripsi ini akan memiliki metodologi sebagai berikut:

1. Melakukan studi literatur tentang metode OIR, teknik ekstraksi fitur lokal SIFT dan ORB, serta teknik-teknik *data mining* yang digunakan pada skripsi ini. Studi literatur dilakukan dengan mencari dan membaca *paper* atau buku yang berkaitan dengan topik tersebut.
2. Mengumpulkan *dataset* gambar POI yang diperlukan untuk penelitian dan pembuatan model identifikasi.
3. Melakukan analisis pada latar belakang masalah pengenalan POI, dengan melihat sifat-sifat fitur lokal pada gambar POI.
4. Menyusun rancangan perangkat lunak.
5. Melakukan implementasi perangkat lunak.
6. Menguji kinerja perangkat lunak.
7. Menulis buku skripsi.

### 1.6 Sistematika Pembahasan

Sistematika pembahasan yang digunakan pada penelitian ini adalah sebagai berikut:

1. Bab 1 Pendahuluan  
Bab ini berisi tentang hal-hal yang menggambarkan skripsi ini secara garis besar. Hal yang dibahas merupakan latar belakang masalah, rumusan masalah, tujuan penelitian, batasan masalah, dan metodologi penelitian.
2. Bab 2 Landasan Teori  
Bab ini berisi tentang dasar-dasar teori dari teknik atau metode yang digunakan dalam skripsi ini, yaitu POI, OIR, metode SIFT, metode ORB, *Best Score Increasing Subsequence* (BSIS), KD-Tree, teknik *clustering Agglomerative* dan DBSCAN, serta metode SIFT dan ORB di *library* OpenCV.
3. Bab 3 Analisis  
Bab ini berisi analisis pada masalah yang dibahas pada skripsi beserta solusi yang digunakan untuk menyelesaikan masalah tersebut.
4. Bab 4 Perancangan  
Bab ini berisi tentang perancangan baik dari metode *clustering* pada pemilihan fitur dan perancangan metode identifikasi POI dengan OIR. Bab juga akan berisi rancangan struktur *file* dan *folder* pada hasil akhir perangkat lunak.
5. Bab 5 Implementasi dan Pengujian  
Bab ini berisi implementasi perangkat lunak pada metode pemilihan fitur dan identifikasi POI serta pengujian terhadap kinerja kedua metode tersebut.

**6. Bab 6 Kesimpulan dan Saran**

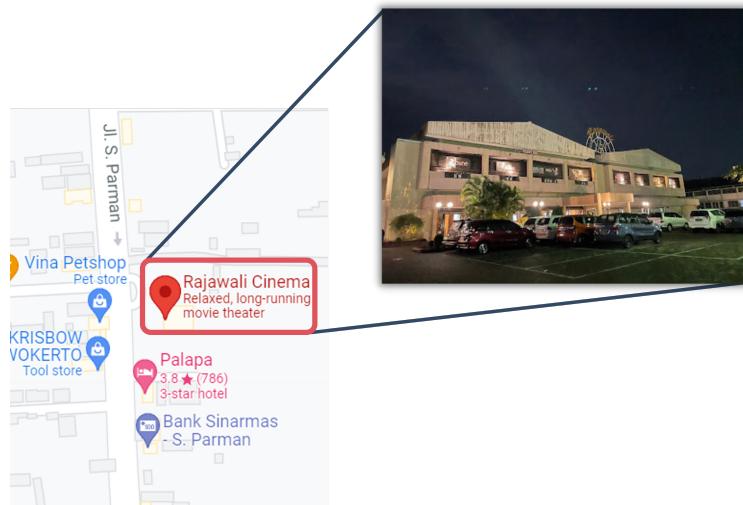
Bab ini berisi kesimpulan yang didapatkan dari hasil analisis serta keseluruhan implementasi dan pengujian yang dilakukan pada penelitian ini.

## BAB 2

### LANDASAN TEORI

#### 2.1 Point of Interest

*Point of Interest* (POI) adalah sebuah lokasi geografis yang memiliki kegunaan tertentu. POI biasanya dikenali oleh banyak orang dan memiliki keunikan tertentu pada tampilannya. Salah satu contoh POI dapat dilihat pada Gambar 2.1. POI juga dapat dimanfaatkan untuk menjadi penanda lokasi seseorang. Seseorang dapat mengerti lokasinya dengan melihat POI yang ada di sekitarnya.



Gambar 2.1: Salah satu contoh POI.

Beberapa POI tertentu dapat memiliki logo yang sifatnya unik. POI tersebut dapat dikenali dengan hanya melihat logonya saja. Contoh bisa dilihat pada Gambar 2.2. POI dengan logo unik ini dapat dikenali oleh komputer salah satunya dengan menggunakan teknik *Object Instance Recognition* (OIR).



Gambar 2.2: Contoh POI dengan logo unik.

## 2.2 Object Instance Recognition

*Object Instance Recognition* (OIR) adalah teknik pengenalan objek spesifik. Pada teknik OIR deteksi tidak memberikan kelas dari gambar tetapi label yang khusus, seperti contohnya jika objek merupakan sebuah mobil, OIR akan memberikan jenis model mobil tersebut. OIR bekerja dengan menerima gambar masukkan dan mencari gambar pada *dataset* yang memiliki paling banyak kemiripan.

OIR dapat dilakukan dengan menggunakan fitur lokal. Fitur lokal sendiri merupakan fitur dalam gambar yang mendeskripsikan sebuah daerah tertentu pada gambar tersebut. Fitur lokal dapat berupa perpotongan garis atau yang biasa disebut *keypoint*. Sebuah *keypoint* akan dapat diidentifikasi dengan menggunakan daerah di sekitar *keypoint* tersebut. Deskripsi *keypoint* ini dibuat dalam sebuah vektor yang dinamakan vektor deskriptor.

Teknik OIR bekerja dengan melakukan deteksi fitur lokal pada gambar masukkan dan pada gambar-gambar di *dataset*. Fitur-fitur lokal dari gambar masukkan tersebut kemudian dipasangkan dengan fitur lokal dari gambar *dataset*. Gambar yang memiliki pasangan terbanyak akan merupakan hasil deteksi gambar masukkan tersebut.

Masalah OIR akan mudah bila gambar masukkan merupakan gambar yang sudah bersih. Pada praktiknya sebuah algoritma OIR seharusnya tetap dapat mengenali objek walaupun gambar bervariasi. Beberapa perubahan pada gambar berikut merupakan faktor-faktor yang dapat mempersulit proses OIR:

- Cahaya  
Perubahan pada tingkat pencahayaan pada gambar yang mengakibatkan perubahan nilai *pixel-pixel* pada gambar.
- Skala  
Jarak diambilnya gambar yang berisi objek. Perbedaan ukuran objek pada gambar akan menyebabkan sudut-sudut pada objek menjadi berbeda.
- Rotasi  
Orientasi atau arah pengambilan gambar yang berbeda akan mengakibatkan objek pada gambar jadi terlihat berbeda. Sudut-sudut akan menjadi berbeda karena arah hadapnya berbeda.
- Latar Belakang  
Objek-objek lain di sekitar objek yang ingin diidentifikasi akan berpotensi mempersulit pemrosesan. Objek-objek tersebut dapat menghasilkan fitur-fitur lokal yang tidak relevan terhadap objek yang ingin diidentifikasi.
- Bagian Objek Tertutup  
Adanya objek lain yang menutupi sebagian dari objek yang ingin diidentifikasi akan berpotensi menyebabkan beberapa fitur lokal dari objek tidak terdeteksi.
- Sudut Pandang  
Sudut pengambilan gambar yang berbeda akan memengaruhi pemrosesan. Fitur-fitur lokal dari objek yang ingin diidentifikasi akan menjadi berbeda.
- Translasi  
Posisi objek yang ingin diidentifikasi dalam gambar akan memengaruhi pemrosesan. Pencarian pasangan fitur lokal tidak dapat dengan hanya menggunakan posisi di mana fitur lokal tersebut ditemukan pada gambar.

Pengambilan fitur lokal dari gambar dapat dilakukan dengan beberapa metode, seperti SIFT (lihat 2.3) dan ORB (lihat 2.4). Kedua metode tersebut—SIFT dan ORB—sudah menangani masalah perubahan skala dan rotasi karena fitur lokal yang dihasilkan oleh SIFT dan ORB bersifat invarian terhadap skala dan rotasi.

Proses pencarian pasangan fitur lokal dari gambar masukkan dan *dataset* dilakukan dengan menggunakan vektor deskriptor dari fitur lokal. Perubahan-perubahan pada gambar walaupun sedikit akan menyebabkan perubahan nilai pada vektor deskriptor. Vektor deskriptor dari fitur

lokal pada gambar masukkan hampir tidak akan persis sama dengan vektor deskriptor dari gambar yang ada di *dataset*. Oleh karena itu pencarian pasangan fitur lokal dilakukan dengan mencari pasangan fitur lokal yang memiliki nilai kemiripan paling tinggi.

Secara garis besar tahapan proses OIR pada penelitian ini adalah sebagai berikut:

1. Ekstraksi Fitur

Pertama akan dilakukan pengambilan fitur-fitur lokal dari gambar masukkan. Dengan menggunakan metode SIFT atau ORB pengambilan fitur lokal akan menghasilkan *list* yang berisi *keypoint*. Setiap *keypoint* yang dihasilkan akan memiliki sebuah vektor yang akan digunakan untuk mengidentifikasi *keypoint* tersebut.

2. Pairing

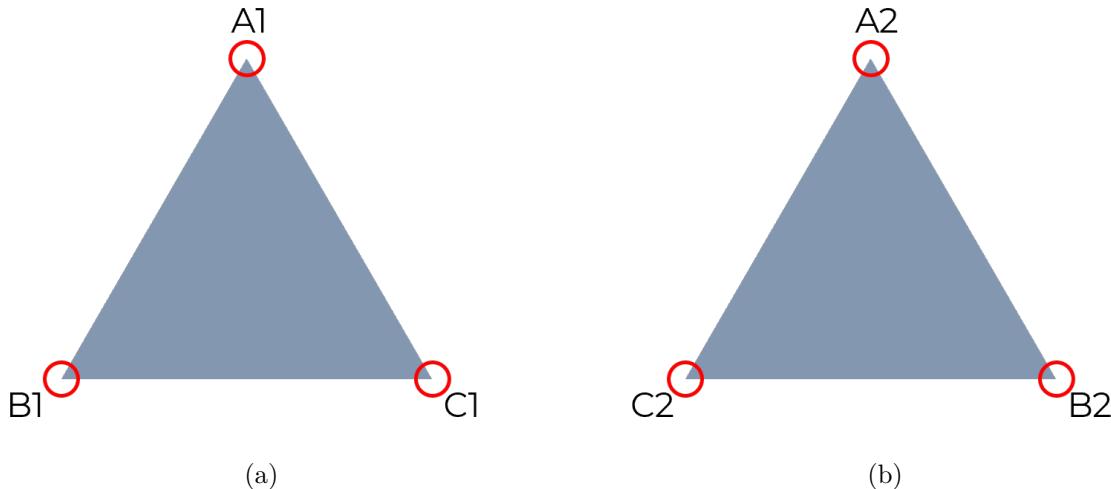
Untuk setiap fitur lokal yang terdeteksi pada gambar masukkan akan dicari beberapa fitur lokal dari gambar pada *dataset* yang paling mirip. Kemiripan fitur lokal ditentukan dengan menghitung jarak *Euclidean* antara vektor deskriptor kedua fitur lokal tersebut.

3. Verification

Pasangan fitur lokal yang didapat pada tahap sebelumnya merupakan pasangan yang hanya memiliki ciri yang mirip tetapi belum tentu konsisten secara geometris. Pasangan fitur lokal dikatakan konsisten secara geometris jika keduanya memiliki posisi yang sama relatif terhadap objek di sekitarnya. Ilustrasi dapat dilihat pada Gambar 2.3. Hanya pasangan-pasangan fitur lokal yang konsisten secara geometris yang akan diproses lebih lanjut.

4. Scoring

Setelah didapatkan semua pasangan fitur lokal yang paling mirip dan konsisten secara geometris maka dapat ditentukan pasangan gambar yang menjadi label gambar masukkan. Kemudian perlu dihitung nilai kemiripan dari label yang dihasilkan. Kemiripan dapat dihitung dengan menghitung  $\frac{1}{d}$  untuk semua pasangan fitur lokal, di mana  $d$  merupakan jarak *Euclidean* pasangan tersebut. Nilai-nilai  $\frac{1}{d}$  tersebut kemudian dihitung totalnya untuk menjadi nilai kemiripan label hasil.



Gambar 2.3: Ilustrasi fitur lokal yang konsisten dan tidak konsisten. Pasangan fitur lokal A1 dengan A2 konsisten secara geometris karena keduanya memiliki posisi yang sama relatif terhadap fitur lokal lain. Fitur lokal A1 berada di atas B1 dan C1 sedangkan fitur lokal A2 berada di atas B2 dan C2. Pasangan fitur lokal B1 dengan B2 dan C1 dengan C2 tidak konsisten secara geometris karena B1 berada di kanan C1 sedangkan B2 berada di sebelah kiri C2.

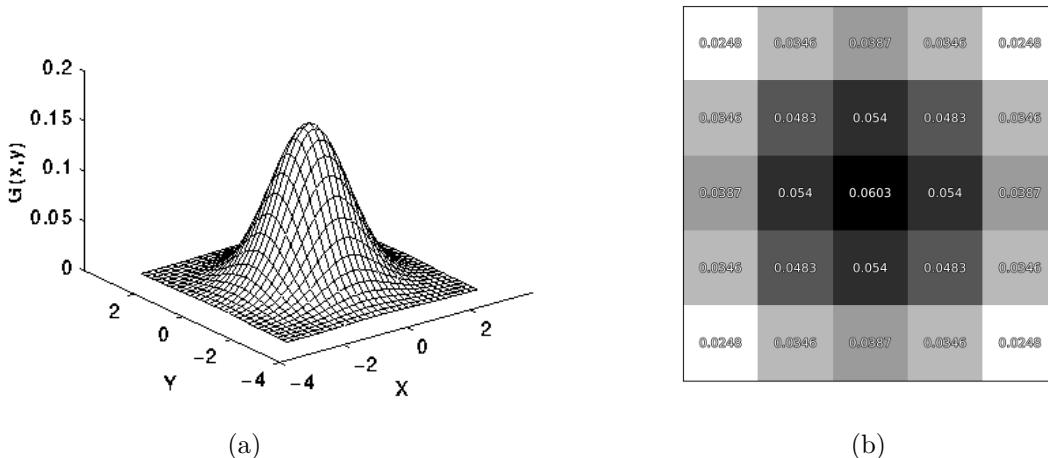
## 2.3 SIFT (Speeded Up Robust Feature)

SIFT adalah salah satu metode pencarian fitur lokal yang dicetuskan pada [1]. Fitur lokal yang dihasilkan SIFT bersifat invarian terhadap rotasi, perubahan skala, dan translasi pada gambar. Sifat invarian ini berarti fitur lokal yang sama pada gambar yang telah di rotasi, diubah skalanya, atau di translasi akan tetap memiliki ciri yang mirip. Setiap fitur lokal akan memiliki sebuah vektor yang mendeskripsikan daerah area fitur lokal tersebut, vektor ini biasa disebut sebagai deskriptor. Vektor deskriptor SIFT berbentuk vektor bilangan bulat yang memiliki 128 elemen. Tahap pencarian fitur lokal pada SIFT dapat dibagi menjadi 4 langkah yang akan dijabarkan pada subbab-subbab berikut.

### 2.3.1 Pencarian Extrema

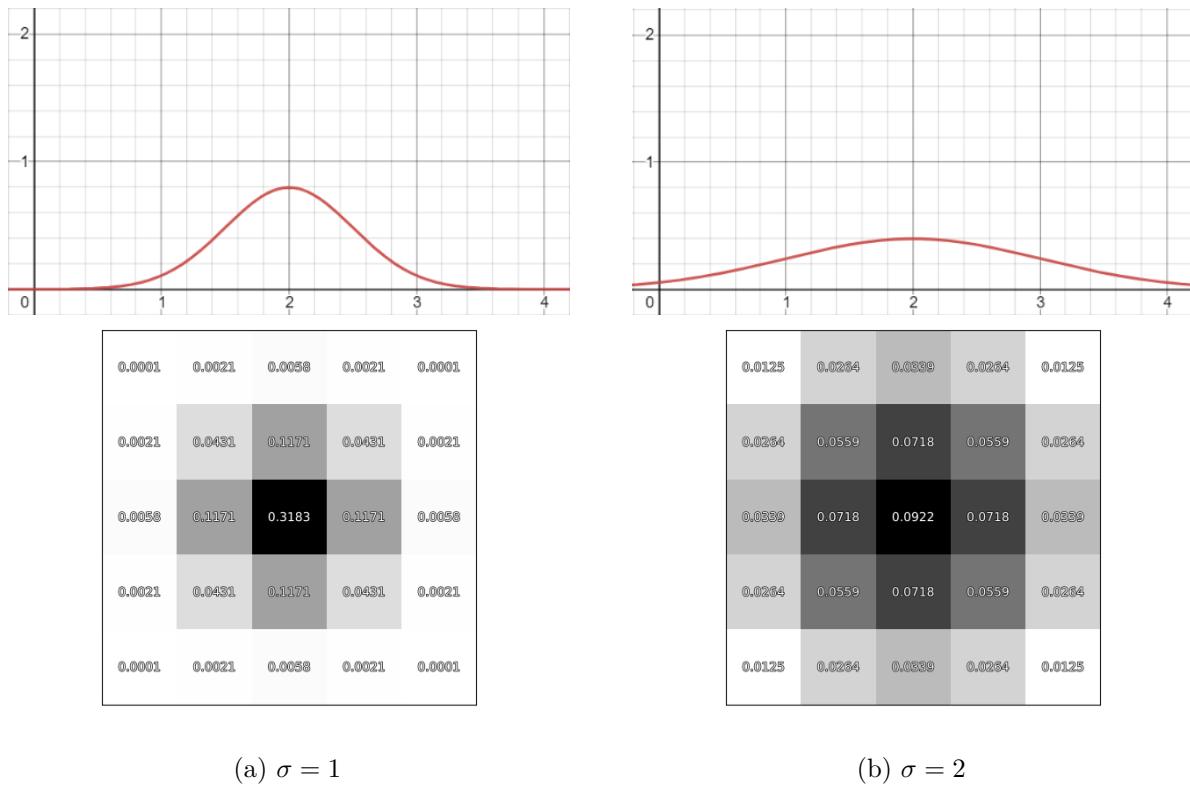
Pada tahap ini akan dicari *pixel-pixel* pada gambar yang merupakan *corner* atau biasa disebut *keypoint*. *Keypoint* pada SIFT dicari dengan memeriksa *pixel-pixel* pada gambar hasil turunan kedua Gaussian. Turunan kedua Gaussian akan dihitung dengan menggunakan *Difference of Gaussian* (DoG).

Penghitungan DoG ini dilakukan dengan memanfaatkan sifat dari *Matrix Konvolusi Gaussian*. *Matrix Konvolusi Gaussian* merupakan *matrix* yang memiliki sifat distribusi Gaussian, di mana titik tengah *matrix* memiliki nilai yang tinggi dan nilai-nilai di sekitarnya semakin mendekati tepi *matrix*. Seperti ditunjukkan pada Gambar 2.4.

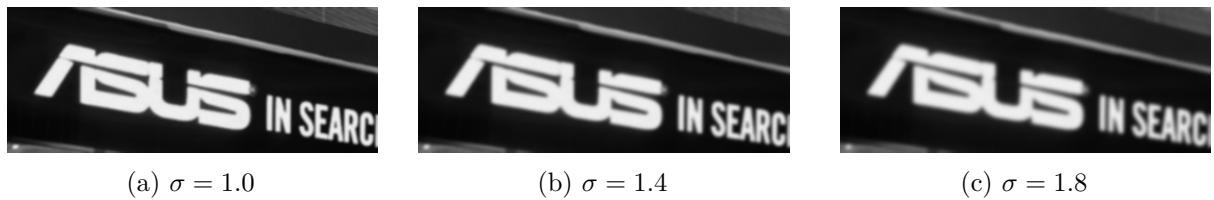


Gambar 2.4: Kurva Gaussian dan bentuk representasi *matrix*-nya.

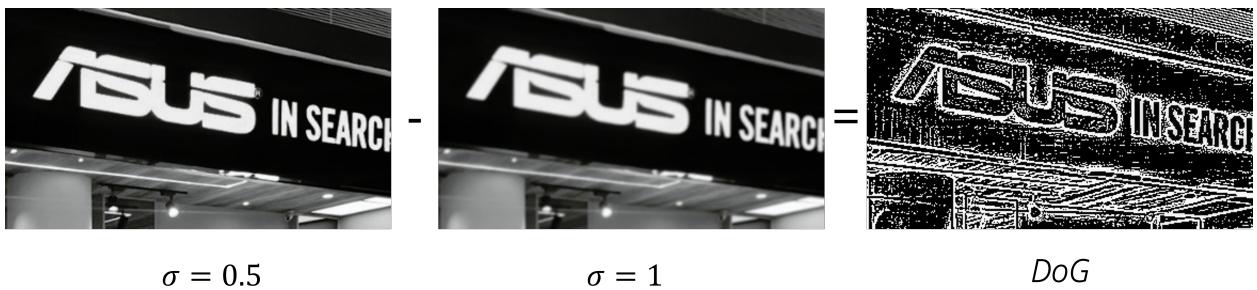
Pada fungsi Gaussian tingkat penyebaran data dapat diatur dengan mengubah parameter  $\sigma$  yang mengatur nilai standar deviasi. Gambar 2.4a menunjukkan bagaimana nilai  $\sigma$  mengatur bagaimana data tersebar dari *mean*, di mana semakin tinggi nilai  $\sigma$  maka data akan semakin menyebar. Nilai yang semakin menyebar menyebabkan perbedaan nilai antar titik semakin kecil. Efeknya pada *matrix* dapat dilihat pada Gambar 2.5. Nilai  $\sigma$  yang tinggi menyebabkan kurva semakin melebar dan pada *matrix* selisih nilai antar titik menjadi semakin kecil.

(a)  $\sigma = 1$ (b)  $\sigma = 2$ Gambar 2.5: Kurva dan *matrix* Gaussian pada nilai  $\sigma$  yang berbeda.

*Matrix* Konvolusi Gaussian ketika diaplikasikan pada gambar akan menyebabkan perubahan nilai tiap *pixel* pada gambar. Nilai dari setiap *pixel* akan menjadi mirip dengan *pixel* tetangga di dekatnya. Perubahan nilai *pixel* akan paling berpengaruh pada daerah dengan perubahan nilai *pixel* yang tinggi. Tingkat perubahan nilai dipengaruhi oleh nilai  $\sigma$  yang digunakan, nilai  $\sigma$  yang tinggi akan menyebabkan nilai *pixel* yang berdekatan semakin mirip—perubahan nilai *pixel* pada daerah tersebut semakin mengecil. Jika dilihat pada gambar, maka gambar hasil konvolusi akan terlihat kabur (*blur*). Nilai  $\sigma$  menentukan tingkat *blur* gambar.

(a)  $\sigma = 1.0$ (b)  $\sigma = 1.4$ (c)  $\sigma = 1.8$ Gambar 2.6: Efek nilai  $\sigma$  pada hasil gambar konvolusi.

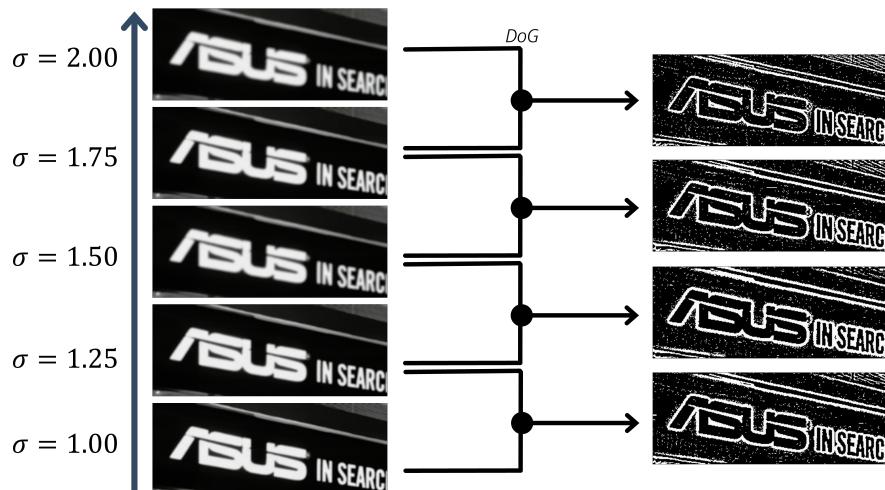
Perubahan nilai  $\sigma$  pada *matrix* konvolusi serta efeknya pada gambar akan dimanfaatkan untuk menghitung *Difference of Gaussian* (DoG). DoG merupakan hasil turunan kedua Gaussian pada gambar. Gambar DoG dapat diperoleh dengan menghitung perbedaan nilai tiap *pixel* dari dua gambar yang telah dikonvolusi oleh *matrix* Gaussian dengan nilai  $\sigma$  yang berbeda. Perbedaan nilai untuk DoG dihitung dengan mengurangi setiap *pixel* pada gambar konvolusi yang memiliki nilai  $\sigma$  yang lebih kecil, dengan setiap *pixel* pada posisi yang sama pada gambar konvolusi yang memiliki nilai  $\sigma$  yang lebih besar. Ilustrasi dapat dilihat pada Gambar 2.7.



Gambar 2.7: Operasi DoG pada gambar

Metode SIFT mencari *keypoint* dengan memanfaatkan konsep DoG. Sebuah gambar akan dikonvolusi dengan *matrix Gaussian* beberapa kali dengan nilai  $\sigma$  yang berbeda. Setelah didapatkan beberapa gambar maka akan dihitung DoG untuk setiap gambar yang nilai  $\sigma$ -nya bersebelahan (Gambar 2.8). Pasangan gambar konvolusi yang berbeda akan menghasilkan gambar DoG yang berbeda juga.

Untuk setiap gambar DoG akan ditentukan *pixel* mana saja yang merupakan *keypoint* dengan mencari *pixel* yang merupakan *extrema*. Sebuah *pixel* merupakan *extrema* jika nilai pixel tersebut lebih besar dari seluruh 26 *pixel* di sekitarnya atau lebih kecil dari seluruhnya. Ke-26 *pixel* tersebut merupakan 8 *pixel* yang mengelilingi, 9 *pixel* pada posisi yang sama dari gambar di atasnya, dan juga 9 *pixel* pada posisi yang sama dari gambar di bawahnya.



Gambar 2.8: Penggunaan DoG pada SIFT

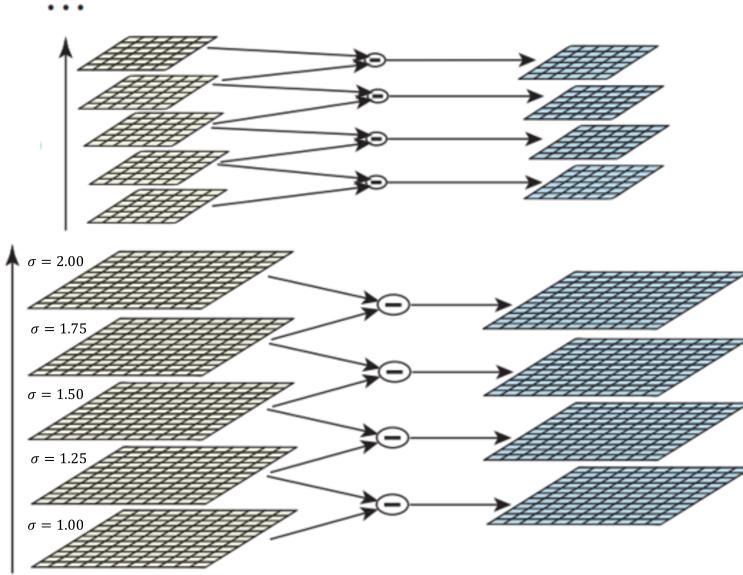
### 2.3.2 Penentuan Skala

Pada tahap sebelumnya sudah didapatkan *keypoint-keypoint* dalam gambar. Agar *keypoint* dapat invariant terhadap skala, *keypoint* perlu untuk dapat tetap terdeteksi walaupun ukuran gambar berubah. Untuk setiap *keypoint* perlu untuk dicari skala terkecil di mana *keypoint* tersebut dapat terdeteksi. Untuk mencapai ini SIFT menggunakan lanjutan dari metode pada Gambar 2.8 dengan langkah sebagai berikut (ilustrasi pada Gambar 2.9):

1. Lakukan konvolusi sampai nilai  $\sigma$  sudah mencapai 2 kali nilai awal
2. Perkecil ukuran gambar (*downsample*) menjadi setengahnya
3. Kembalikan nilai  $\sigma$  ke nilai awal
4. Ulang tahap dari langkah 1 hingga gambar sudah terlalu kecil.

Pada langkah di atas setiap siklus ukuran gambar disebut sebagai oktaf, dimulai dari oktaf pertama, lalu kedua, dan seterusnya. Dengan setiap oktaf ukuran gambar akan semakin kecil.

Pencarian *keypoint* dilakukan pada tiap oktaf, dan untuk tiap *keypoint* tersebut ditulis nilai oktaf tertinggi (ukuran gambar terkecil) di mana *keypoint* tersebut dapat terdeteksi.



Gambar 2.9: Oktaf pada proses konvolusi SIFT

### 2.3.3 Penentuan Orientasi

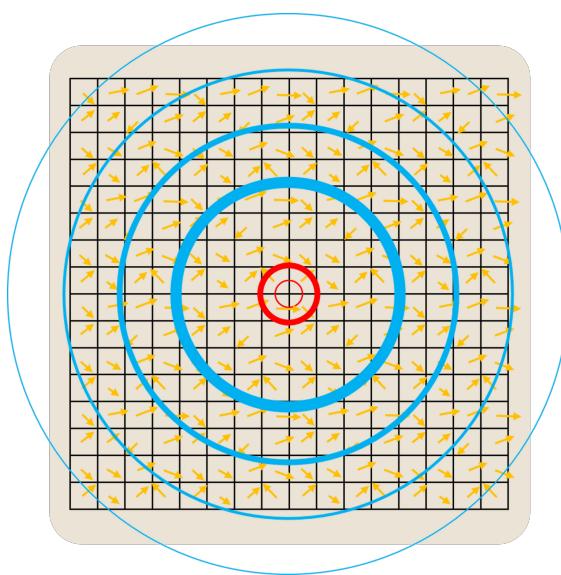
Untuk dapat invariant terhadap rotasi gambar, setiap *keypoint* perlu memiliki orientasi yang konsisten. Untuk mendapatkan orientasi yang sama pada setiap rotasi gambar, orientasi perlu ditentukan dari atribut yang akan selalu sama bagaimanapun gambar dirotasi. Untuk itu orientasi *keypoint* ditentukan dengan menggunakan orientasi yang dominan dari *pixel-pixel* di sekitar *keypoint*. Luas daerah yang digunakan untuk mendapat orientasi ditentukan oleh skala dari *keypoint*.

Penentuan orientasi yang dominan dihitung dengan menggunakan *magnitude*,  $m(x, y)$ , dan orientasi,  $\theta(x, y)$ , dari *pixel-pixel* dengan menggunakan rumus berikut,  $L(x, y)$  merupakan gambar hasil konvolusi:

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2} \quad (2.1)$$

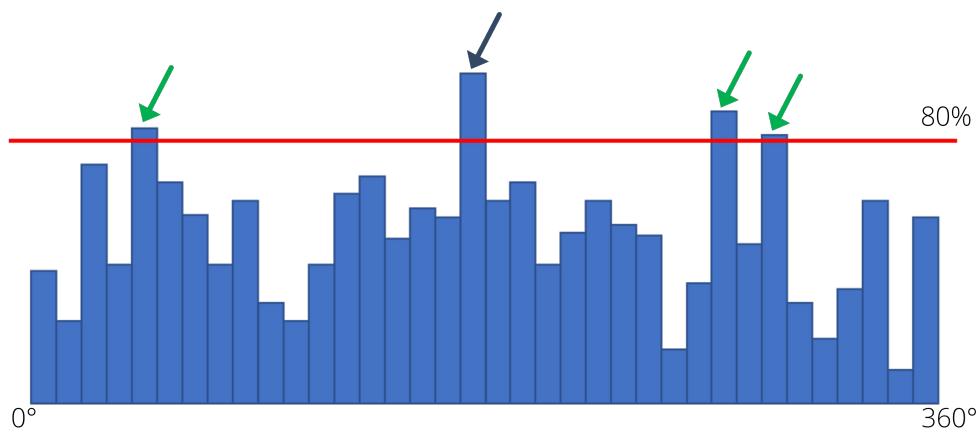
$$\theta(x, y) = \tan^{-1}((L(x, y+1) - L(x, y-1)) / (L(x+1, y) - L(x-1, y))) \quad (2.2)$$

Setiap *pixel* akan dihitung orientasi dan *magnitude*-nya. *Magnitude* akan digunakan sebagai bobot dari *pixel* tersebut. Selain *magnitude*, bobot sebuah *pixel* juga dipengaruhi oleh *Gaussian Weighting*. *Pixel* yang posisinya dekat dengan titik pusat (pusat *keypoint*) akan memiliki bobot yang lebih tinggi dibanding yang lokasinya jauh. Ilustrasi pada Gambar 2.10 menunjukkan bagaimana pembobotan dihitung, *pixel-pixel* yang berada dekat dengan *keypoint* (titik tengah) akan diberi bobot yang lebih besar—ditandai dengan lingkaran yang tebal. Sedangkan bobot akan semakin berkurang untuk *pixel* yang jauh dari *keypoint*.



Gambar 2.10: Ilustrasi pembobotan pada *Gaussian Weighting*. Titik tengah merupakan *keypoint* yang diperiksa sedangkan setiap kotak merupakan *pixel-pixel* di sekitar *keypoint*. Tanda panah pada tiap kotak menunjukkan *magnitude* dan orientasi *pixel* tersebut, panjang panah merupakan nilai *magnitude* dan arahnya merupakan orientasi

Setelah setiap *pixel* sudah dihitung orientasi dan bobotnya—menggunakan *magnitude* dan *Gaussian Weighting*—nilai bobot tersebut akan dimasukkan ke dalam histogram berdasarkan orientasinya. Histogram yang digunakan memiliki 36 bin yang masing-masing mewakili 10 derajat orientasi. Ilustrasi dapat dilihat pada Gambar 2.11.



Gambar 2.11: Histogram untuk menentukan orientasi dari *keypoint*. Bin dengan nilai tertinggi (tanda panah biru) akan digunakan sebagai orientasi dari *keypoint*. Untuk bin lain yang jumlahnya berada dalam rentang 80% dari bin tertinggi (tanda panah hijau) digunakan untuk membuat *keypoint* baru.

Dari histogram tersebut puncak nilai bin tertinggi akan digunakan sebagai orientasi dari *keypoint*. Untuk puncak-puncak lain yang berada dalam rentang 80% dari puncak tertinggi akan digunakan untuk membuat *keypoint* baru pada lokasi yang sama dengan orientasi yang berbeda sesuai dengan nilai orientasi pada bin tersebut.

### 2.3.4 Pembuatan Deskriptor

Setelah didapatkan *keypoint* beserta skala dan orientasinya, perlu untuk diberikan sebuah identitas pada setiap *keypoint*. Pemberian identitas ini berguna untuk mengidentifikasi *keypoint* yang satu dengan yang lainnya, agar dapat ditemukan *keypoint-keypoint* dengan ciri yang sama. Identifikasi *keypoint* ditentukan dengan membuat sebuah vektor deskriptor, yaitu vektor yang mendeskripsikan daerah di sekitar *keypoint*. Vektor deskriptor pada SIFT berbentuk vektor sepanjang 128 bilangan bulat.

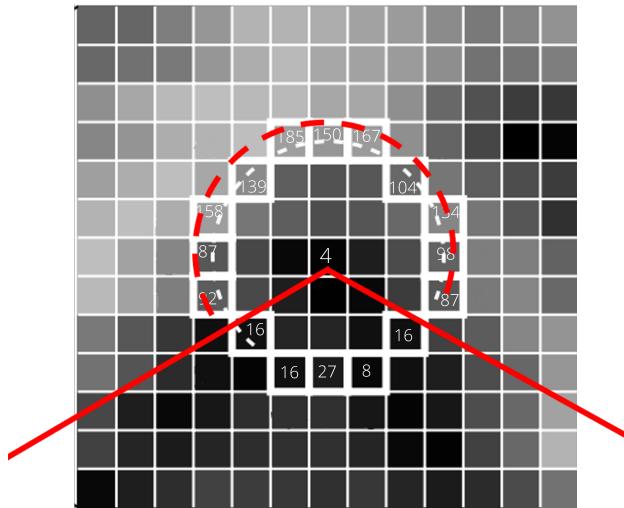
Pembuatan vektor dilakukan dengan mengambil daerah di sekitar *keypoint* dari gambar yang terlebih dahulu dirotasi sesuai dengan orientasi *keypoint*, ukuran daerah berdasarkan pada skala. Daerah tersebut kemudian dibagi menjadi  $4 \times 4$  subdaerah. Untuk setiap subdaerah dihitung nilai *magnitude* dan orientasi setiap *pixel*-nya dengan diberi bobot menggunakan *Gaussian Weighting* lalu hasilnya dimasukkan ke dalam histogram dengan 8 bin. Setiap bin dalam histogram mewakili 45 derajat orientasi, jumlah dari setiap bin ini akan dijadikan nilai pada vektor deskriptor. Teradapat total 16 subdaerah dengan setiap daerah menghasilkan 8 bilangan, sehingga didapat total sebanyak  $16 \times 8 = 128$  elemen untuk vektor deskriptor.

## 2.4 ORB (Oriented FAST and Rotated BRIEF)

ORB adalah metode pencarian fitur lokal yang dijelaskan pada [2]. ORB dapat menemukan fitur lokal dengan lebih cepat jika dibandingkan dengan SIFT, walaupun fitur lokal yang dihasilkan tidak seakurat yang dihasilkan SIFT. ORB mencari fitur lokal dengan mencari *pixel* yang merupakan *keypoint*. *Keypoint* dalam ORB dicari dengan ide bahwa sebuah *pixel* yang merupakan sudut akan memiliki daerah kontinu dengan nilai intensitas yang lebih kecil atau lebih besar dari nilai intensitas *pixel* tersebut. Fitur lokal yang dihasilkan ORB akan memiliki sebuah vektor deskriptor yang berbentuk vektor biner sebanyak 256 elemen. Tahap pencarian fitur lokal pada ORB dibagi menjadi 4 langkah yang dijelaskan pada subbab-subbab berikut.

### 2.4.1 Pencarian Keypoint

Untuk menentukan apakah sebuah *pixel* dalam gambar merupakan *keypoint*, ORB mengambil nilai *pixel* tersebut dan 16 *pixel* di sekitarnya yang membentuk lingkaran. Dari ke 16 *pixel* tersebut dibandingkan nilainya dengan *pixel* yang di tengah  $p$ , yaitu *pixel* yang ingin diperiksa apakah merupakan *keypoint*. Sebuah *pixel* merupakan *keypoint* jika dari 16 *pixel* di sekitarnya terdapat setidaknya  $n$  *pixel* kontinu yang nilainya lebih besar dari nilai  $p + t$  atau lebih kecil dari  $p - t$ . Proses ini diilustrasikan pada Gambar 2.12

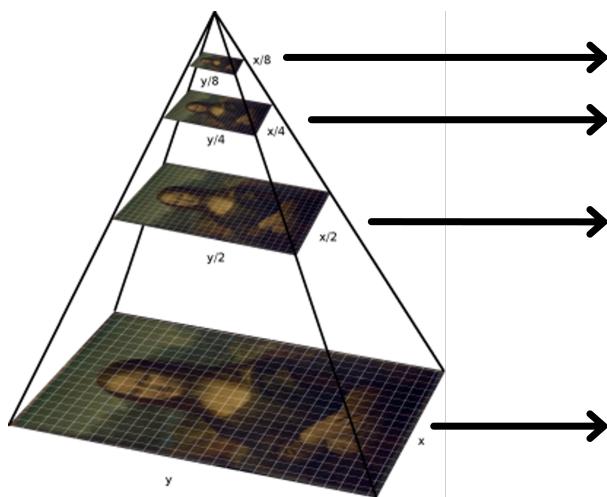


Gambar 2.12: Ilustrasi *keypoint* pada ORB. Titik tengah pada gambar tersebut memiliki *pixel-pixel* dengan nilai jauh lebih kecil (lebih gelap) yang mengelilinginya. *Pixel-pixel* yang mengelilinginya tersebut seakan membentuk sudut dengan titik tengah sebagai pusatnya.

#### 2.4.2 Penentuan Skala

*Keypoint* yang telah dideteksi pada tahap awal perlu dapat terdeteksi juga walaupun ukuran gambar berubah agar sifatnya invariant terhadap skala. Untuk mencapai sifat ini ORB menggunakan metode *Image Pyramid*. ORB menggunakan *Image Pyramid* dengan cara memperkecil ukuran gambar beberapa kali dan untuk setiap ukuran gambar dilakukan deteksi untuk *keypoint*.

Ilustrasi dapat dilihat pada Gambar 2.13. Pada ilustrasi tersebut gambar awal diperkecil beberapa kali dengan membagi panjang dan lebarnya menjadi setengahnya. Untuk setiap ukuran gambar dicari *keypoint-keypoint*-nya.



Gambar 2.13: *Image Pyramid* pada ORB

#### 2.4.3 Penentuan Orientasi

Orientasi *keypoint* pada ORB ditentukan oleh arah hadap titik *keypoint* pada titik *Intensity Centroid* di daerah sekitarnya. *Intensity Centroid* pada sebuah daerah gambar merupakan titik di mana terjadi perubahan nilai intensitas terbesar. Titik *Intensity Centroid*,  $C$ , didefinisikan sebagai berikut:

$$C = \left( \frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right) \quad (2.3)$$

*Centroid* ditentukan dengan menghitung *moment* pada gambar yang didefinisikan sebagai berikut:

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y) \quad (2.4)$$

Titik *centroid* dihitung pada daerah yang dikelilingi 16 *pixel* yang digunakan pada tahap Penentuan *Keypoint*. Dari daerah tersebut didapat titik yang merupakan *centroid*. Lalu dibuat garis vektor yang berasal dari titik *keypoint* (titik tengah) menuju titik *centroid*. Orientasi ditentukan dari sudut antara garis lurus sumbu x dengan garis vektor.

#### 2.4.4 Pembuatan Deskriptor

### 2.5 BSIS (Best Score Increasing Subsequence)

Pada tahapan OIR (lihat 2.2) sebuah pasangan fitur lokal perlu untuk memiliki sifat yang mirip (dilihat dari vektor deskriptornya) dan juga konsisten secara geometris. Pasangan yang konsisten secara geometris adalah pasangan yang memiliki posisi spasial yang konsisten terhadap objek (fitur lokal) di sekitarnya. Salah satu metode yang dapat digunakan untuk menentukan apakah pasangan fitur lokal bersifat konsisten secara geometris adalah BSIS. Seluruh isi bab ini didapatkan dari [3]

BSIS dilakukan dengan memberikan modifikasi metode OIR. Perubahan terdapat pada tahap *Pairing*, *Verification*, dan *Scoring*. Langkah-langkah BSIS secara rinci dijelaskan pada subbab-subbab berikut, dilakukan setelah dilakukan ekstraksi fitur lokal pada gambar masukkan:

#### 2.5.1 Pairing

Pertama untuk setiap fitur lokal pada gambar masukkan akan dicari  $N$  fitur lokal dari *dataset* yang nilai vektor deskriptornya paling mirip atau memiliki jarak yang *euclidean* paling kecil. BSIS mengasumsikan jarak *euclidean* tiap pasangan sebuah fitur lokal tersebut secara normal.

Pasangan-pasangan yang sudah dihitung lalu di-*filter* dengan mengambil hanya nilai yang merupakan *outlier* di sisi kiri. Pasangan yang digunakan hanya merupakan pasangan yang nilainya kurang dari  $m - (K \times \sigma)$ ,  $m$  merupakan *mean* dan  $\sigma$  merupakan standar deviasi dengan nilai  $K = 4$ .

Setelah itu untuk setiap pasangan akan diberi bobot. Bobot,  $P_w$ , dihitung dengan  $P_w = \frac{distance(P_Q, P_T) - m}{\sigma}^2$ ,  $P_Q$  merupakan fitur lokal gambar masukkan dan  $P_T$  merupakan fitur lokal dari *dataset*. Bobot tersebut pada dasarnya menghitung sejauh apa nilai jarak pasangan tersebut terhadap rata-ratanya. Karena pasangan yang digunakan pada tahap ini hanya pasangan yang nilai jaraknya lebih kecil dari rata-rata maka nilai bobot menunjukkan tingkat kemungkinan pasangan tersebut merupakan pasangan yang benar.

#### 2.5.2 Verification

Pada tahap ini akan dilakukan pemeriksaan pada pasangan-pasangan fitur lokal untuk mencari pasangan mana saja yang konsisten secara geometris. Pasangan yang digunakan adalah pasangan yang sudah di-*filter* pada tahap sebelumnya dan diberi bobot. Setiap fitur lokal dari gambar masukkan dapat dipasangkan dengan lebih dari satu fitur lokal dari gambar *dataset* di pasangan yang berbeda. Langkah untuk melakukan verifikasi adalah sebagai berikut:

1. Pada gambar masukkan, berikan label pada tiap fitur lokal berdasarkan urutan kemunculannya dari sumbu x. Label urutan ini disebut *order*.
2. Urutkan fitur lokal pada gambar *dataset* berdasarkan sumbu x dan masukkan ke dalam kolom-kolom. Untuk kolom tulis semua pasangan dari fitur lokal tersebut dan beri *order* dari fitur lokal gambar masukkan pasangan tersebut. Catat juga  $P_w$  dari tiap pasangan
3. Dari urutan kolom fitur lokal gambar *dataset* buat sebuah *subsequence* yang memaksimalkan nilai total  $P_w$ . *Subsequence* dibuat dengan aturan berikut:
  - Dari tiap kolom hanya dapat diambil satu pasangan.

- *Order* dari pasangan pada *subsequence* harus selalu meningkat.
4. Simpan pasangan-pasangan yang membentuk *subsequence* ke dalam  $V_x$ .
  5. Ulangi lagi tahapan dengan menggunakan sumbu y pada pasangan-pasangan di  $V_x$ . Untuk lebih lanjut lagi mencari pasangan yang tidak konsisten dari sumbu y.

Tahapan di atas dilakukan beberapa kali pada rotasi fitur lokal gambar *dataset* yang berbeda untuk mencari rotasi gambar yang memberikan paling banyak pasangan fitur lokal yang konsisten.

### 2.5.3 Scoring

Setelah dipilih dan didapat pasangan fitur lokal yang konsisten secara geometris akan dihitung nilai kemiripan gambar masukkan dengan gambar *dataset* yang terpilih. Tingkat kemiripan ditentukan dengan menghitung total  $P_w$  dari pasangan yang terpilih pada tahap *Verification*. Nilai kemiripan ini dapat digunakan untuk menentukan apakah pasangan gambar cukup mirip untuk menjadi pasangan yang benar.

## 2.6 KD-Tree

## 2.7 Clustering

*Clustering* adalah salah satu teknik pengolahan data dalam *machine learning*. Pada dasarnya *clustering* akan membagi objek menjadi beberapa kelompok (*cluster*) berdasarkan sifatnya. Objek yang memiliki sifat mirip akan masuk kedalam satu kelompok. Sebuah pembagian *cluster* yang baik adalah di mana setiap objek dalam *cluster* memiliki sifat yang mirip dan antar *cluster* memiliki sifat yang sangat berbeda.

Beberapa contoh metode *clustering* yang ada adalah *Agglomerative* dan *DBSCAN*. Kedua teknik tersebut menggunakan metode dasar yang berbeda dan akan menghasilkan *cluster* dengan ciri yang berbeda juga. Kedua metode tersebut dijelaskan pada dua subbab berikut. Penjelasan berdasarkan pada [4]

### 2.7.1 Agglomerative

Teknik *clustering Agglomerative* adalah salah satu teknik *clustering* yang berbasis hierarki (*hierarchical*). Teknik ini membentuk *cluster* dengan menyusun hierarki antar objek berdasarkan kemiripannya. *Agglomerative* adalah teknik *clustering hierarchical* yang menggunakan metode *bottom-up*. Tahapan dimulai dengan membentuk *cluster-cluster* kecil dan kemudian menggabungkan *cluster* kecil tersebut menjadi *cluster* yang lebih besar.

Tahapan *clustering* pada *Agglomerative* dimulai dengan terlebih dahulu menghitung jarak *euclidean* antar tiap objek. Setelah itu ambil pasangan dengan jarak paling kecil dan gabungkan menjadi satu *cluster* dan lakukan juga untuk jarak paling kecil berikutnya. Jika jarak terkecil yang ada adalah antara objek yang sudah berada di dalam *cluster*, maka gabungkan kedua *cluster* tersebut menjadi *cluster* yang lebih besar.

Langkah-langkah pada tahapan tersebut dilakukan hingga semua objek sudah berada di dalam satu *cluster* yang sama. *Cluster* besar tersebut lalu dapat dibagi berdasarkan dari jaraknya dengan menggunakan *threshold* yang dapat ditentukan secara manual.

### 2.7.2 DBSCAN

*DBSCAN* atau *Density-Based Spatial Clustering of Applications with Noise* adalah salah satu metode *clustering* yang berbasis kepadatan (*density based*). Pada *DBSCAN* sebuah *cluster* merupakan sebuah daerah padat yang dipisahkan oleh daerah yang jarang. *Cluster* yang dihasilkan tidak selalu berbentuk *circular* dan objek-objek yang berada dalam satu *cluster* tidak selalu merupakan data

yang mirip. Karena bentuknya yang tidak *circular* dan tidak tentu miripnya objek, *cluster* tidak dapat direpresentasikan dengan sebuah titik tengah atau *centroid*.

Penyusunan *cluster* pada DBSCAN dimulai dengan mencari objek yang merupakan *core object*. *Core object* merupakan objek yang memiliki setidaknya *MinPts* objek lain pada radius  $\epsilon$  yang berpusat pada objek tersebut. *MinPts* dan  $\epsilon$  adalah parameter yang ditentukan secara manual. Nilai *MinPts* dan  $\epsilon$  akan memengaruhi hasil *cluster* yang dihasilkan.

Setiap objek yang merupakan *core object* beserta anggotanya (objek lain pada radius  $\epsilon$ ) akan menjadi satu *cluster*. Jika dalam radius  $\epsilon$  objek tersebut terdapat objek lain yang juga merupakan *core object*, maka akan digabungkan menjadi satu *cluster*. *Core object* yang saling berdekatan ini akan terus digabungkan menjadi *cluster* yang besar.



## **DAFTAR REFERENSI**

- [1] Lowe, G. (2004) Sift-the scale invariant feature transform. *Int. J.*, **2**, 2.
- [2] Rublee, E., Rabaud, V., Konolige, K., dan Bradski, G. (2011) Orb: An efficient alternative to sift or surf. *2011 International conference on computer vision*, pp. 2564–2571. Ieee.
- [3] Kusuma, G. P., Harjono, K. D., dan Putra, M. T. D. (2019) Geometric verification method of best score increasing subsequence. *IEEE* , ?
- [4] Han, J., Pei, J., dan Kamber, M. (2011) *Data mining: concepts and techniques*. Elsevier.



# LAMPIRAN A

## KODE PROGRAM

Kode A.1: MyCode.c

```

1 // This does not make algorithmic sense,
2 // but it shows off significant programming characters.
3
4 #include<stdio.h>
5
6 void myFunction( int input, float* output ) {
7     switch ( array[1] ) {
8         case 1: // This is silly code
9             if ( a >= 0 || b <= 3 && c != x )
10                 *output += 0.005 + 20050;
11             char = 'g';
12             b = 2^n + ~right_size - leftSize * MAX_SIZE;
13             c = ( --aaa + &daa ) / ( bbb++ - ccc % 2 );
14             strcpy(a,"hello_$.@");
15     }
16     count = ~mask | 0x00FF00AA;
17 }
18
19 // Fonts for Displaying Program Code in LATEX
20 // Adrian P. Robson, nepswb.co.uk
21 // 8 October 2012
22 // http://nepswb.co.uk/docs/progfonts.pdf
23

```

Kode A.2: MyCode.java

```

1 import java.util.ArrayList;
2 import java.util.Collections;
3 import java.util.HashSet;
4
5 //class for set of vertices close to furthest edge
6 public class MyFurSet {
7     protected int id;                                //id of the set
8     protected MyEdge FurthestEdge;                   //the furthest edge
9     protected HashSet<MyVertex> set;                //set of vertices close to furthest edge
10    protected ArrayList<ArrayList<Integer>> ordered; //list of all vertices in the set for each trajectory
11    protected ArrayList<Integer> closeID;           //store the ID of all vertices
12    protected ArrayList<Double> closeDist;          //store the distance of all vertices
13    protected int totaltrj;                          //total trajectories in the set
14
15    /*
16     * Constructor
17     * @param id : id of the set
18     * @param totaltrj : total number of trajectories in the set
19     * @param FurthestEdge : the furthest edge
20     */
21    public MyFurSet(int id,int totaltrj,MyEdge FurthestEdge) {
22        this.id = id;
23        this.totaltrj = totaltrj;
24        this.FurthestEdge = FurthestEdge;
25        set = new HashSet<MyVertex>();
26        ordered = new ArrayList<ArrayList<Integer>>();
27        for (int i=0;i<totaltrj;i++) ordered.add(new ArrayList<Integer>());
28        closeID = new ArrayList<Integer>(totaltrj);
29        closeDist = new ArrayList<Double>(totaltrj);
30        for (int i = 0;i <totaltrj;i++) {
31            closeID.add(-1);
32            closeDist.add(Double.MAX_VALUE);
33        }
34    }
35}
36

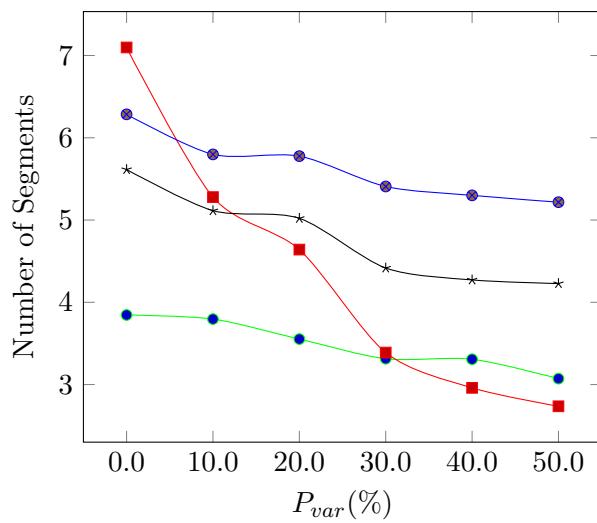
```



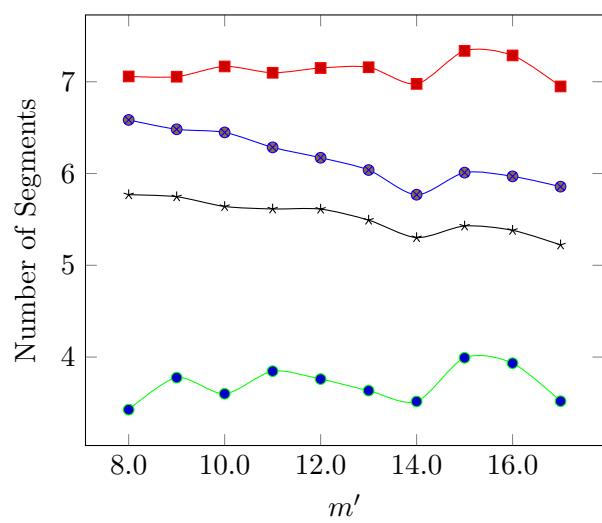
## LAMPIRAN B

### HASIL EKSPERIMENT

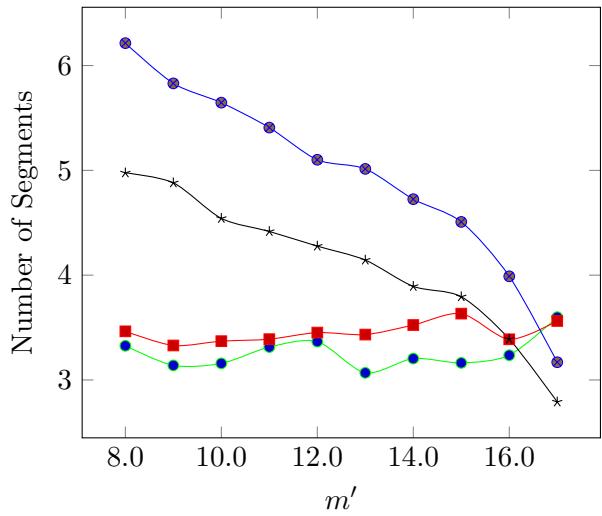
Hasil eksperimen berikut dibuat dengan menggunakan TIKZPICTURE (bukan hasil excel yg diubah ke file bitmap). Sangat berguna jika ingin menampilkan tabel (yang kuantitasnya sangat banyak) yang datanya dihasilkan dari program komputer.



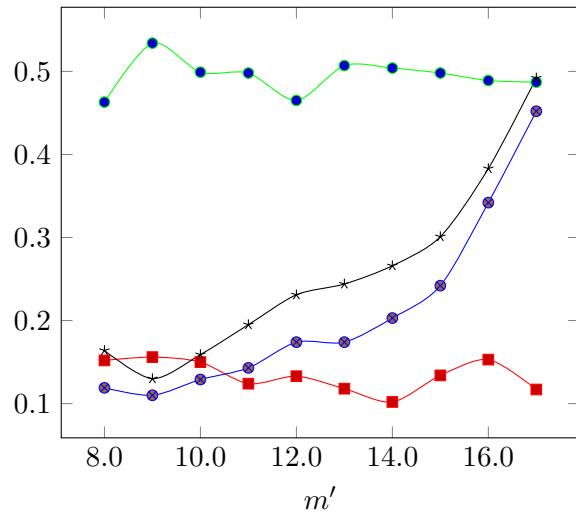
Gambar B.1: Hasil 1



Gambar B.2: Hasil 2



Gambar B.3: Hasil 3



Gambar B.4: Hasil 4