**COMPUTER VISION AND IMAGE PROCESSING**

**Percobaan 6: Neighborhood Operator (Morfologi dan Distance Transform)**



**Oleh:**

**Luthfi Aminulloh 1020181013**

**TEKNIK ELEKTRO**

**PROGRAM PASCASARJANA S2 TERAPAN**

**POLITEKNIK ELEKTRONIKA NEGERI SURABAYA**

**2019**

# Tujuan Percobaan

1. Mahasiswa mengetahui operasi morfologi pada citra dan memahami fungsinya
2. Mahasiswa dapat menggunakan beberapa operator morfologi untuk melakukan rekayasa pada citra

# Persiapan

1. Praktikum ini dapat dikerjakan dengan pra-syarat bahwa mahasiswa:

* Telah mendapatkan matematika lanjut
* Memiliki konsep dasar sinyal dan sistem
* Memiliki konsep dasar pemrograman dan pengoperasian MS Visual C++

1. Software yang diperlukan:

* Microsoft Visual C++ 2010 express atau lebih tinggi.
* OpenCV Library 2.4.9 atau lebih tinggi.

1. Sarana penunjang praktikum:

* File gambar (.bmp, .jpg)
* File video (.avi)

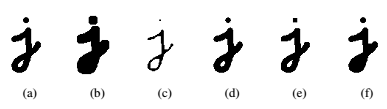
# Pendahuluan

Pada praktikum kali ini, akan dibahas dua macam operasi neighborhood yang terdiri dari dua operator, yaitu:

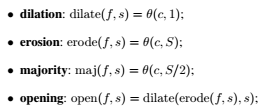
* Operasi morfologi
* Distance transform

Operasi **morfology** adalah sebuah operasi pada citra biner untuk mengubah bentuk dari obyek biner, mengkonvolusikan citra biner dengan sebuah binary structuring element, serta memilih nilai output biner tergantung pada hasil threshold dari konvolusi. Beberapa operasi morphology:

* Dilation (b)
* Erosion (c)
* Majority (d)
* Opening (e)
* Closing (f)



Operasi standar yang digunakan dalam morphology biner adalah:



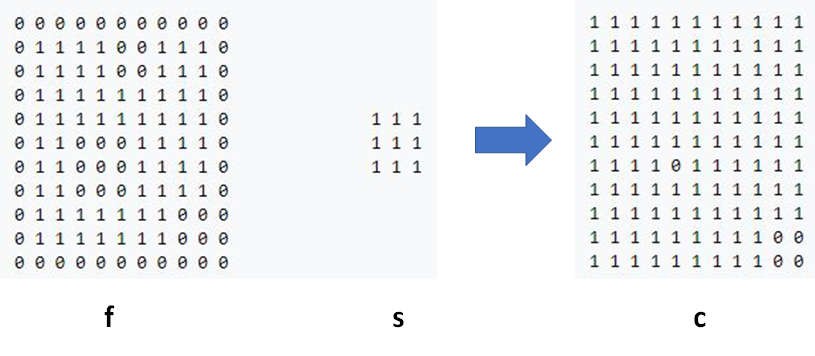
Dimana:

f : gambar masukan

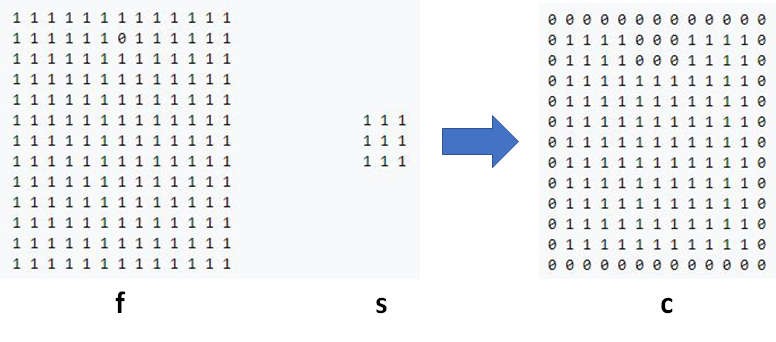
s : structuring element

c : gambar hasil

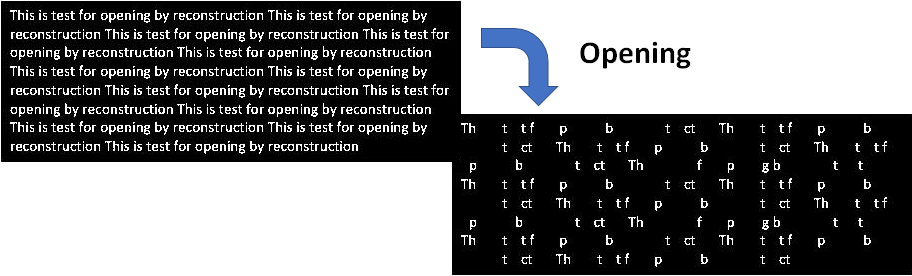
Dilasi adalah operasi morfologi yang akan menambahkan piksel pada batasan dari objek dalam suatu image sehingga nantinya apabila dilakukan operasi ini maka image hasilnya lebih besar ukurannya dibandingkan dengan image aslinya.



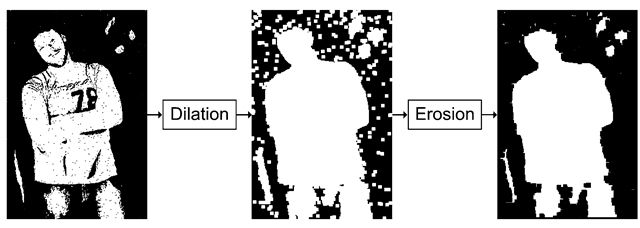
Erosi adalah operasi morphology yang akan menghilangkan piksel pada batasan-batasan objek yang akan di erosi. Jumlah dari piksel yang dihilangkan bergantung pada ukuran dan bentuk dari structuring element yang digunakan untuk memproses image tersebut.



Opening adalah operasi morphology yang merupakan kombinasi proses dimana suatu citra digital dikenai operasi erosi dilanjutkan dengan dilasi. Operasi opening pada citra mempunyai efek memperhalus batas-batas objek, memisahkan objek-objek yang sebelumnya bergandengan, dan menghilangkan objek-objek yang lebih kecil daripada ukuran structuring.



Closing adalah pperasi morphology yang merupakan kombinasi dimana suatu citra dikenai operasi dilasi dilanjutkan dengan erosi. Operasi closing juga cenderung akan memperhalus objek pada citra, namun dengan cara menyambung pecahan-pecahan (fuses narrow breaks and thin gulf) dan menghilangkan lubang-lubang kecil pada objek.



Sedangkan **Distance Transform** D(i,j) dari sebuah gambar biner b(i,j) didefinisikan sebagai berikut. Katakanlah d(k,l) adalah beberapa matriks jarak antara offset piksel. Dua pengukuran yang umum digunakan meliputi city block atau Manhattan distance dan Euclidean distance.

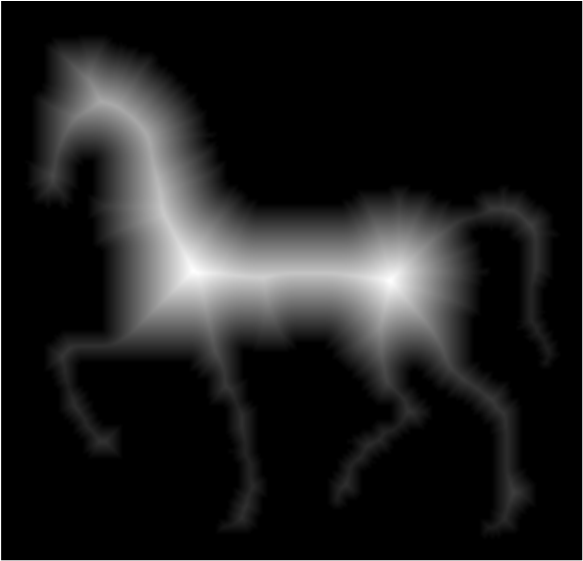
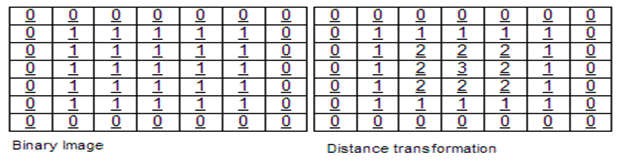
d1(k, l) = |k| + |l| d2(k, l) = √(k2 + l2)

Distance Transform kemudian didefinisikan sebagai

D(i, j) = min d(i − k, j − l)

Dimana k, l: b(k, l) = 0

Persamaan diatas adalah jarak piksel terhadap background terdekat dimana nilai adalah 0.



# Prosedur percobaan

## Dilasi

|  |
| --- |
| List Program: |
| #include <math.h>  #include <string.h>  #include <opencv2\core\core.hpp>  #include <opencv2\highgui\highgui.hpp>  #include <opencv2\imgproc\imgproc.hpp>  using namespace cv;  //Percobaan 1 Dilasi  int main()  {  Mat source, destination;  source = imread("../data/morph.jpg", CV\_LOAD\_IMAGE\_COLOR);  // Create a structuring element  int dilation\_size = 3; // Ukuran operator dilasi  Mat element = getStructuringElement(cv::MORPH\_CROSS,  cv::Size(2 \* dilation\_size + 1, 2 \* dilation\_size +1),  cv::Point(dilation\_size, dilation\_size) );  // Apply dilation on the image  dilate(source,destination,element);  namedWindow("Display window", CV\_WINDOW\_AUTOSIZE);  imshow("Display window", source);  namedWindow("Result window", CV\_WINDOW\_AUTOSIZE);  imshow("Result window", destination);  waitKey(0);  return 0;  } |

Hasil Percobaan:

|  |
| --- |
|  |
|  |

Langkah Percobaan:

* Pada contoh program diatas, ukuran structuring element dilasi yang digunakan adalah 3x3. Ubah-ubahlah ukuran structuring element tersebut berturut-turut 7x7, 11x11, dan 15x15! Amati apakah yang terjadi dengan gambar hasilnya? Jelaskan!

|  |  |  |
| --- | --- | --- |
| 7x7 | 11x11 | 15x15 |
|  |  |  |

* Ubahlah tipe dilasi “**cv::MORPH\_CROSS**” dengan tipe dilasi yang lain, yaitu “**cv::MORPH\_RECT**” dan “**cv::MORPH\_ELLIPSE**”! Amati, apakah yang terjadi?

|  |  |
| --- | --- |
| **MORPH\_RECT** | **MORPH\_ELLIPSE** |
|  |  |

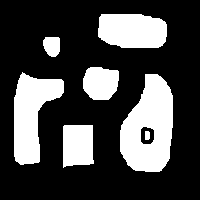
* Tambahkan sebuah looping “***for****”* sebelum proses dilasi untuk melakukan pengulangan jumlah dilasi, misalnya: “***for(i=0;i<N;i++)***”, dimana N dapat dimasukkan sebelum proses dilasi dimulai! Cobalah untuk nilai N=1, N=5 dan N=10! Amati, apakah yang terjadi?

|  |
| --- |
|  |
| int main()  {  Mat source, destination;  source = imread("../data/morph.jpg", CV\_LOAD\_IMAGE\_COLOR);  // Create a structuring element  int dilation\_size = 3; // Ukuran operator dilasi  Mat element = getStructuringElement(cv::MORPH\_ELLIPSE,  cv::Size(2 \* dilation\_size + 1, 2 \* dilation\_size +1),  cv::Point(dilation\_size, dilation\_size) );    // Apply dilation on the image  Mat temp;  Mat temp1;  temp = source;  int N = 5;  for (size\_t i = 0; i < N; i++)  {  dilate(temp, temp1, element);  temp1 = temp;  }  namedWindow("Source", CV\_WINDOW\_AUTOSIZE);  imshow("Source", source);  namedWindow("Result", CV\_WINDOW\_AUTOSIZE);  imshow("Result", temp1);  waitKey(0);  return 0;  } |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

* Buatlah sebuah gambar berukuran 200 x 200 piksel menggunakan program “paint”, misalnya:



* Dengan menggunakan program yang sudah anda modifikasi pada nomor 3, proseslah gambar diatas tersebut. Apakah yang terjadi? Jelaskan!

## Erosi

|  |
| --- |
| Listing Program: |
| int main()  {  Mat source, destination;  source = imread("../data/OrangeCat.jpg", CV\_LOAD\_IMAGE\_COLOR);  // Create a structuring element  int erosion\_size = 3; // Ukuran operator erosi  Mat element = getStructuringElement(cv::MORPH\_CROSS,  cv::Size(2 \* erosion\_size + 1, 2 \* erosion\_size + 1),  cv::Point(erosion\_size, erosion\_size) );  // Apply erosion on the image  erode(source,destination,element);  namedWindow("Display window", CV\_WINDOW\_AUTOSIZE);  imshow("Display window", source);  namedWindow("Result window", CV\_WINDOW\_AUTOSIZE);  imshow("Result window", destination);  waitKey(0);  return 0;  } |

Hasil Percobaan:

|  |
| --- |
|  |
|  |

Langkah Percobaan:

* Pada contoh program diatas, ukuran structuring element erosi yang digunakan adalah 3x3. Ubah-ubahlah ukuran structuring element tersebut berturut-turut 7x7, 11x11, dan 15x15! Amati apakah yang terjadi dengan gambar hasilnya? Jelaskan!

|  |  |  |
| --- | --- | --- |
| 7x7 | 11x11 | 15x15 |
|  |  |  |

* Ubahlah tipe erosi “**cv::MORPH\_CROSS**” dengan tipe erosi yang lain, yaitu “**cv::MORPH\_RECT**” dan “**cv::MORPH\_ELLIPSE**”! Amati, apakah yang terjadi?

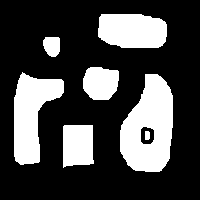
|  |  |
| --- | --- |
| **MORPH\_RECT** | **MORPH\_ELLIPSE** |
|  |  |

* Tambahkan sebuah looping “***for****”* sebelum proses erosi untuk melakukan pengulangan jumlah erosi, misalnya: “***for(i=0;i<N;i++)***”, dimana N dapat dimasukkan sebelum proses erosi dimulai! Cobalah untuk nilai N=1, N=5 dan N=10! Amati, apakah yang terjadi?

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

* Buatlah sebuah gambar berukuran 200 x 200 piksel menggunakan program “paint”, misalnya:



Dengan menggunakan program yang sudah anda modifikasi pada nomor 3, proseslah gambar diatas tersebut. Apakah yang terjadi? Jelaskan!

## Opening and Closing

Langkah Percobaan:

1. Berdasarkan pada dua eksperimen sebelumnya, buatlah sebuah program opening!

|  |
| --- |
|  |
| int main() {  Mat source1, destination1, destination2;  source1 = imread("../data/morph.jpg", CV\_LOAD\_IMAGE\_COLOR);  // Create a structuring element  int erosion\_size = 3; // Ukuran operator erosi  Mat element1 = getStructuringElement(cv::MORPH\_ELLIPSE,  cv::Size(2 \* erosion\_size + 1, 2 \* erosion\_size + 1),  cv::Point(erosion\_size, erosion\_size));  // Apply erosion on the image  erode(source1, destination1, element1);  // Create a structuring element  int dilation\_size = 3; // Ukuran operator dilasi  Mat element2 = getStructuringElement(cv::MORPH\_ELLIPSE,  cv::Size(2 \* dilation\_size + 1, 2 \* dilation\_size + 1),  cv::Point(dilation\_size, dilation\_size));  // Apply dilation on the image  dilate(destination1, destination2, element2);  namedWindow("Source", CV\_WINDOW\_AUTOSIZE);  imshow("Source", source1);  namedWindow("Erode", CV\_WINDOW\_AUTOSIZE);  imshow("Erode", destination1);  namedWindow("Dilate", CV\_WINDOW\_AUTOSIZE);  imshow("Dilate", destination2);  waitKey(0);  return 0;  } |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

1. Set ukuran structuring element opening yang digunakan 3x3. Ubah-ubahlah ukuran structuring element tersebut berturut-turut 7x7, 11x11, dan 15x15! Amati apakah yang terjadi dengan gambar hasilnya? Jelaskan!

|  |
| --- |
|  |
| Dimensi 7x7 |

|  |
| --- |
|  |
| Dimensi 11x11 |

|  |
| --- |
|  |
| Dimensi 15x15 |

1. Ubahlah tipe dilasi dan erosi “**cv::MORPH\_CROSS**” dengan tipe dilasi dan erosi yang lain, yaitu “**cv::MORPH\_RECT**” dan “**cv::MORPH\_ELLIPSE**”! Amati, apakah yang terjadi?

|  |
| --- |
|  |
| **MORPH\_RECT** |

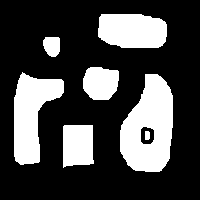
|  |
| --- |
|  |
| **MORPH\_ELLIPSE** |

1. Tambahkan sebuah looping “***for****”* sebelum proses opening untuk melakukan pengulangan jumlah opening, misalnya: “***for(i=0;i<N;i++)***”, dimana N dapat dimasukkan sebelum proses opening dimulai! Cobalah untuk nilai N=1, N=5 dan N=10! Amati, apakah yang terjadi?

|  |
| --- |
|  |
| N=5 |

|  |
| --- |
|  |
| N=10 |

1. Buatlah sebuah gambar berukuran 200 x 200 piksel menggunakan program “paint”, misalnya:



Dengan menggunakan program yang sudah anda modifikasi pada nomor 4, proseslah gambar diatas tersebut. Apakah yang terjadi? Jelaskan!

1. Sekarang buatlah sebuah program closing!

|  |
| --- |
| Listing Program: |
| //Program Clossing  int main() {  Mat source1, destination1,destination2;  source1 = imread("../data/morph.jpg", CV\_LOAD\_IMAGE\_COLOR);  // Create a structuring element  int dilation\_size = 3; // Ukuran operator dilasi  Mat element1 = getStructuringElement(cv::MORPH\_ELLIPSE,  cv::Size(2 \* dilation\_size + 1, 2 \* dilation\_size + 1),  cv::Point(dilation\_size, dilation\_size));  // Apply dilation on the image  dilate(source1, destination1, element1);  // Create a structuring element  int erosion\_size = 3; // Ukuran operator erosi  Mat element2 = getStructuringElement(cv::MORPH\_ELLIPSE,  cv::Size(2 \* erosion\_size + 1, 2 \* erosion\_size + 1),  cv::Point(erosion\_size, erosion\_size));  // Apply erosion on the image  erode(destination1, destination2, element2);  namedWindow("Source", CV\_WINDOW\_AUTOSIZE);  imshow("Source", source1);  namedWindow("Dilate", CV\_WINDOW\_AUTOSIZE);  imshow("Dilate", destination1);  namedWindow("Erode", CV\_WINDOW\_AUTOSIZE);  imshow("Erode", destination2);  waitKey(0);  return 0;  } |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Source Image | Dilate Image | Erode Image |

1. Set ukuran structuring element closing yang digunakan 3x3. Ubah-ubahlah ukuran structuring element tersebut berturut-turut 7x7, 11x11, dan 15x15! Amati apakah yang terjadi dengan gambar hasilnya? Jelaskan!

|  |
| --- |
|  |
| Dimensi 7x7 |

|  |
| --- |
|  |
| Dimensi 11x11 |

|  |
| --- |
|  |
| Dimensi 15x15 |

1. Ubahlah tipe dilasi dan erosi “**cv::MORPH\_CROSS**” dengan tipe dilasi dan erosi yang lain, yaitu “**cv::MORPH\_RECT**” dan “**cv::MORPH\_ELLIPSE**”! Amati, apakah yang terjadi?

|  |
| --- |
|  |
|  |

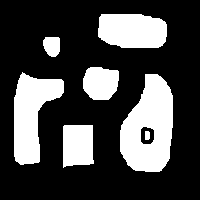
|  |
| --- |
|  |
|  |

1. Tambahkan sebuah looping “***for****”* sebelum proses closing untuk melakukan pengulangan jumlah closing, misalnya: “***for(i=0;i<N;i++)***”, dimana N dapat dimasukkan sebelum proses closing dimulai! Cobalah untuk nilai N=1, N=5 dan N=10! Amati, apakah yang terjadi?

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

1. Buatlah sebuah gambar berukuran 200 x 200 piksel menggunakan program “paint”, misalnya:



Dengan menggunakan program yang sudah anda modifikasi pada nomor 9, proseslah gambar diatas tersebut. Apakah yang terjadi? Jelaskan!

## Distance Transform

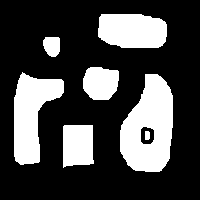
|  |
| --- |
| Listing Program: |
| int main()  {  Mat source, destination;  source = imread("../data/OrangeCat.jpg", CV\_LOAD\_IMAGE\_COLOR);  // Create a structuring element  int erosion\_size = 3; // Ukuran operator erosi  Mat element = getStructuringElement(cv::MORPH\_CROSS,  cv::Size(2 \* erosion\_size + 1, 2 \* erosion\_size + 1),  cv::Point(erosion\_size, erosion\_size) );  // Apply erosion on the image  erode(source,destination,element);  namedWindow("Display window", CV\_WINDOW\_AUTOSIZE);  imshow("Display window", source);  namedWindow("Result window", CV\_WINDOW\_AUTOSIZE);  imshow("Result window", destination);  waitKey(0);  return 0;  } |

Hasil percobaan:

|  |
| --- |
|  |
|  |

Langkah percobaan:

1. Cobalah program diatas! Amati, apa yang terjadi!
2. Buatlah sebuah gambar berukuran 200 x 200 piksel menggunakan program “paint”, misalnya:



Dengan menggunakan program tersebut, proseslah gambar diatas tersebut. Apakah yang terjadi? Jelaskan!

|  |
| --- |
|  |
|  |