PRAKTIKUM 12

Morfologi

Materi

- Operasi-operasi morfologi (dilatasi, erosi, *opening*, *closing*)
- Contoh Implementasi dilatasi dan erosi di OpenCV

Tujuan Praktikum

- Mahasiswa dapat mengimplementasikan operasi-operasi morfologi di OpenCV.

A PENYAJIAN

Morfologi

- Morfologi adalah proses mengidentifikasi bentuk dengan basis region (citra bertipe biner & *grayscale*)
- Tujuannya adalah untuk memperbaiki hasil segmentasi

Operasi-operasi Morfologi yang sering digunakan:

1. Dilatasi
2. Erosi
3. *Opening*
4. *Closing*
5. Dll

Dilasi

Merupakan proses penggabungan titik-latar (0) menjadi bagian dari objek (1), berdasarkan *structuring element* (SE) yang digunakan.

$$D(A,S)=A \oplus S$$

Fungsi Dilasi pada OpenCV

C++: void **dilate**(InputArray **src**, OutputArray **dst**, InputArray **kernel**, Point **anchor**=Point(-1,-1), int **iterations**=1, int **borderType**=BORDER_CONSTANT, const Scalar& **borderValue**=morphologyDefaultBorderValue())

Keterangan:

Parameters:
- **src** – input image; the number of channels can be arbitrary, but the depth should be one of `CV_8U`, `CV_16U`, `CV_16S`, `CV_32F` or ``CV_64F`.
- **dst** – output image of the same size and type as `src`.
- **element** – structuring element used for dilation; if `element=Mat()`, a `3 x 3` rectangular structuring element is used.
- **anchor** – position of the anchor within the element; default value `(-1, -1)` means that the anchor is at the element center.
- **iterations** – number of times dilation is applied.
- **borderType** – pixel extrapolation method (see `borderInterpolate()` for details).
- **borderValue** – border value in case of a constant border (see `createMorphologyFilter()` for details).

Erosi

Merupakan proses penghapusan titik-titik objek (1) menjadi bagian dari latar (0), berdasarkan *structuring element* (SE) yang digunakan.

$$E(A,S)=A \otimes S$$

Fungsi Erosi pada OpenCV

C++: void **erode**(InputArray **src**, OutputArray **dst**, InputArray **kernel**, Point **anchor**=Point(-1,-1), int **iterations**=1, int **borderType**=BORDER_CONSTANT, const Scalar& **borderValue**=morphologyDefaultBorderValue())

Keterangan:

Parameters:
- **src** – input image; the number of channels can be arbitrary, but the depth should be one of `CV_8U`, `CV_16U`, `CV_16S`, `CV_32F` or ``CV_64F`.
- **dst** – output image of the same size and type as `src`.
- **element** – structuring element used for erosion; if `element=Mat()`, a `3 x 3` rectangular structuring element is used.
- **anchor** – position of the anchor within the element; default value `(-1, -1)` means that the anchor is at the element center.
- **iterations** – number of times erosion is applied.
- **borderType** – pixel extrapolation method (see `borderInterpolate()` for details).
- **borderValue** – border value in case of a constant border (see `createMorphologyFilter()` for details).

Opening

- *Opening* adalah proses erosi yang diikuti dengan dilasi.
- Efek yang dihasilkan adalah menghilangnya objek-objek kecil dan kurus, memecah objek pada titik-titik yang kurus, dan secara umum menghaluskan batas objek besar tanpa mengubah area objek secara signifikan
- *Opening* berguna untuk menghaluskan citra, menghilangkan tonjolan yang tipis.

$$Ao\ S = (A \otimes S) \oplus S$$

Closing

- *Closing* adalah proses dilasi yang diikuti dengan erosi.
- Efek yang dihasilkan adalah mengisi lubang kecil pada objek, menggabungkan objek-objek yang berdekatan, dan secara umum menghaluskan batas dari objek besar tanpa mengubah area objek secara signifikan
- Berguna untuk menghaluskan citra, menghilangkan lubang yang kecil.

$$A \bullet S = (A \oplus S) \otimes S$$

B LATIHAN

Morfologi.cpp

Berikut ini merupakan latihan membuat program yang dapat melakukan dilasi, erosi, *noise removal*, *opening*, *closing*, dan *skeletonizing*. (**Download "praktikum 12 – citra-latihan-morfologi.zip" di LMS terlebih dahulu)**

```
#include <iostream>
#include <cv.h>
#include <highgui.h>
#include <math.h>

using namespace cv;
```

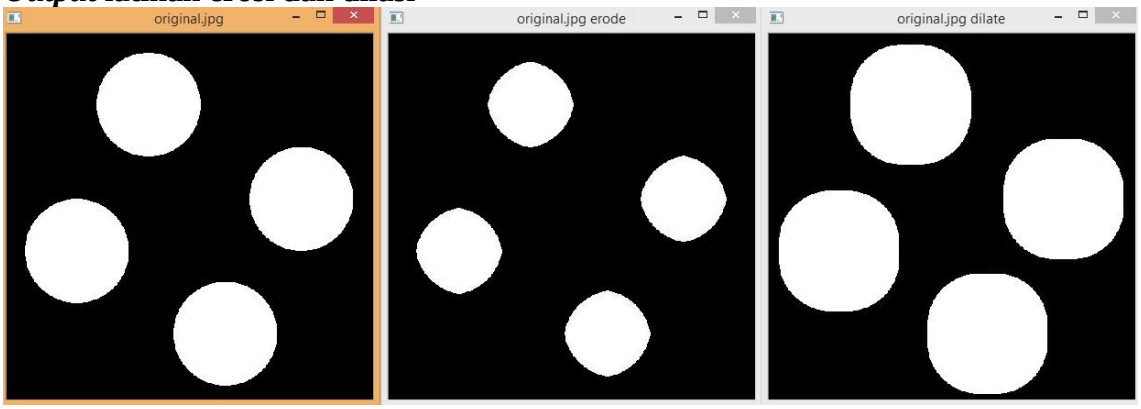
```
using namespace std;

void latihan_erosi_dan_dilasi()
{
    string filename = "original.jpg";
    Mat original = imread(filename, 0);

    int k_size=9;
    Mat element = getStructuringElement( MORPH_RECT,
    Size( 2*k_size + 1, 2*k_size+1 ), Point( k_size, k_size ) );

    Mat erodeImg;
    Mat dilateImg;

    erode(original,erodeImg,element);
    dilate(original,dilateImg,element);

    imshow(filename, original);
    imshow(filename+" erode", erodeImg);
    imshow(filename+" dilate", dilateImg);
    waitKey(0);
}

void latihan_noise_removal()
{
    string filename = "noise.jpg";
    Mat original = imread(filename, 0);

    int k_size=3;
    Mat element = getStructuringElement( MORPH_RECT,
    Size( 2*k_size + 1, 2*k_size+1 ), Point( k_size, k_size ) );

    Mat outputImg;

    erode(original,outputImg,element);
    imshow(filename+" erode", outputImg);
    dilate(outputImg,outputImg,element);

    imshow(filename, original);
    imshow(filename+" noise remove result", outputImg);
    waitKey(0);
}

void latihan_opening()
{
    string filename = "opening.jpg";
    Mat original = imread(filename, 0);

    int k_size=3;
    Mat element = getStructuringElement( MORPH_RECT,
    Size( 2*k_size + 1, 2*k_size+1 ), Point( k_size, k_size ) );

    Mat outputImg;

    erode(original,outputImg,element);
    imshow(filename+" erode (1x)", outputImg);
    erode(outputImg,outputImg,element);
```

```
    imshow(filename+" erode (2x)", outputImg);
    dilate(outputImg,outputImg,element);
    dilate(outputImg,outputImg,element);

    imshow(filename, original);
    imshow(filename+" opening result", outputImg);
    waitKey(0);
}

void latihan_closing()
{
    string filename = "closing.jpg";
    Mat original = imread(filename, 0);

    int k_size=3;
    Mat element = getStructuringElement( MORPH_RECT,
    Size( 2*k_size + 1, 2*k_size+1 ), Point( k_size, k_size ) );

    Mat outputImg;

    dilate(original,outputImg,element);
    imshow(filename+" dilate (1x)", outputImg);
    dilate(outputImg,outputImg,element);
    imshow(filename+" dilate (2x)", outputImg);
    erode(outputImg,outputImg,element);
    erode(outputImg,outputImg,element);

    imshow(filename, original);
    imshow(filename+" closing result", outputImg);
    waitKey(0);
}

void latihan_skeletonizing()
{
    string filename = "original.jpg";
    Mat original = imread(filename, 0);

    int k_size=3;
    Mat element = getStructuringElement( MORPH_RECT,
    Size( 2*k_size + 1, 2*k_size+1 ), Point( k_size, k_size ) );

    Mat erodeImg;
    erode(original,erodeImg,element);
    imshow(filename+" erode", erodeImg);

    //sub original - erodeImg
    Mat output = original.clone();
    output = original-erodeImg;

    imshow(filename, original);
    imshow(filename+" skeletonizing result", output);
    waitKey(0);
}

int main()
{
    latihan_erosi_dan_dilasi();
```

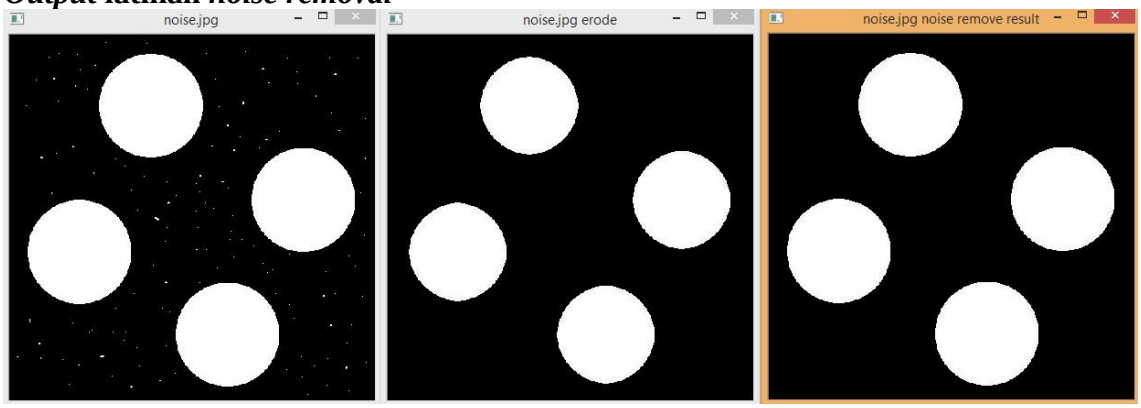
```
    //latihan_noise_removal();
    //latihan_opening();
    //latihan_closing();
    //latihan_skeletonizing();
    return 0;
}
```

Berikut ini merupakan *output* dari morfologi. Cpp

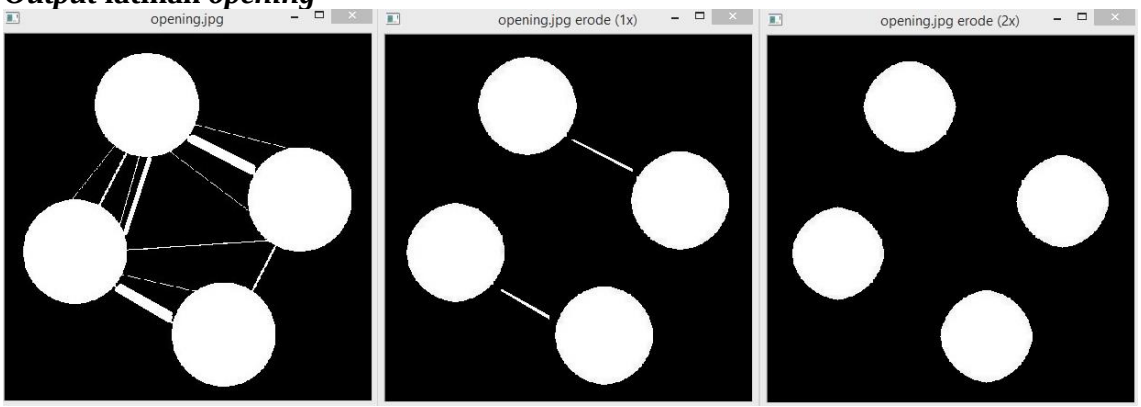
***Output* latihan erosi dan dilasi**

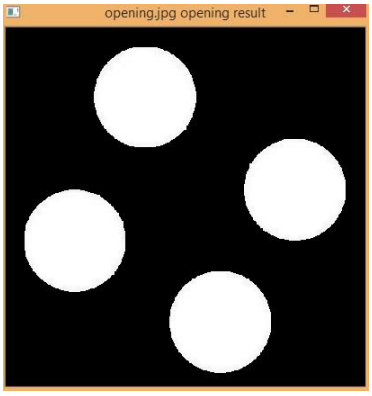


Output* latihan *noise removal

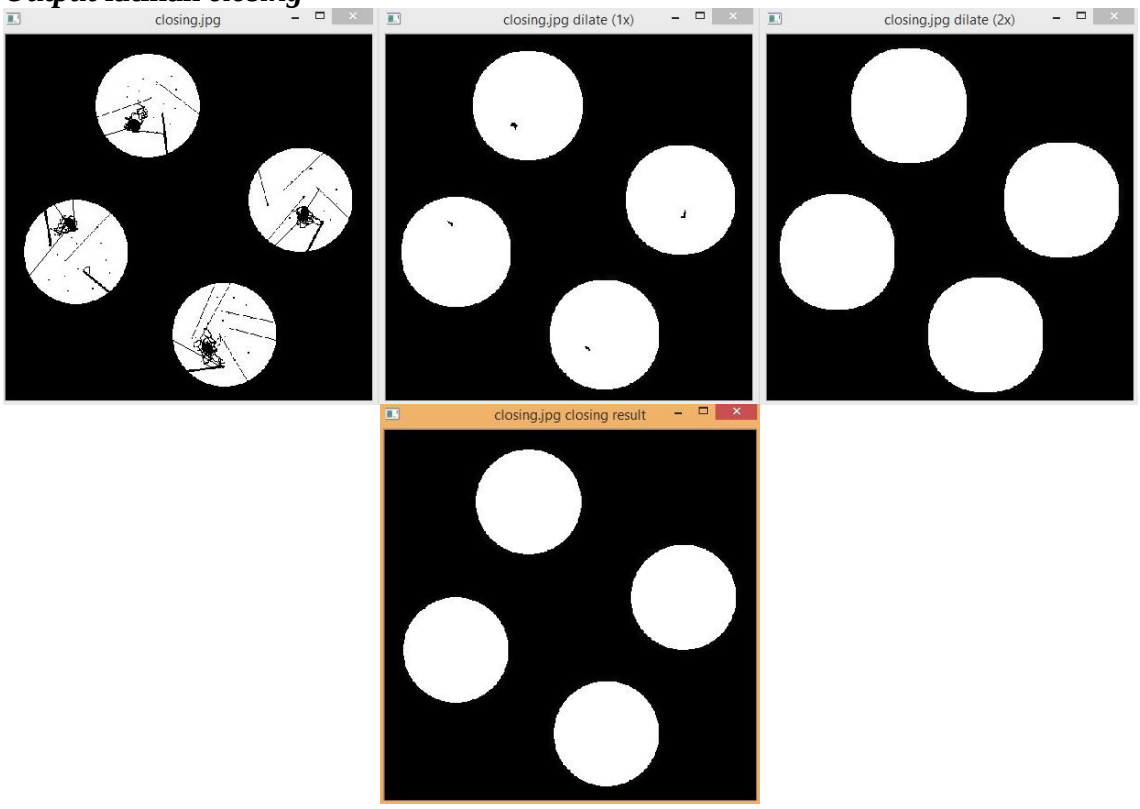


Output* latihan *opening

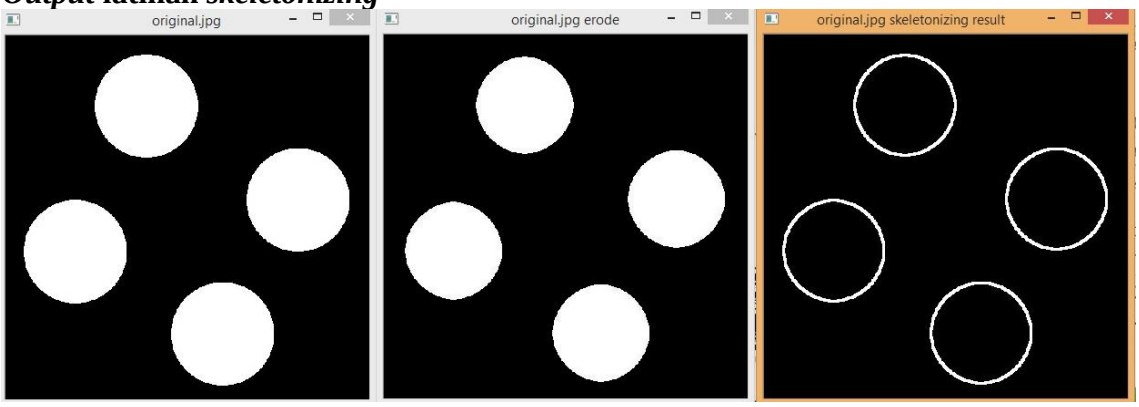




Output* latihan *closing



Output* latihan *skeletonizing



Nama :
NRP :
Nama Dosen :
Nama Asisten :

C TUGAS

Simpan kode program beserta *screenshot* citra hasil segmentasi dan citra hasil *closing*. *File* disimpan dengan format LKP12_NIM_Kelas dalam file .pdf.

Segmentasi dan *closing*

1 *Download* citra tomato.jpg pada LMS LKP12
2 Lakukanlah segmentasi pada tomato.jpg dan *closing* terhadap hasil citra biner segmentasi **(Boleh menggunakan fungsi OpenCV).**
3 Tentukan nilai *threshold,* ukuran *blur,* jumlah operasi dilasi, dan operasi erosi untuk melakukan segmentasi dan dilanjutkan dengan *closing*.
4 Tuliskan alasan singkat terhadap pemilihan ukuran *kernel* dan jumlah operasi dilasi dan erosi yang dilakukan.

Kode program

```
#include <iostream>
#include <cv.h>
#include <highgui.h>
#include <math.h>

using namespace cv;
using namespace std;

void tomato_segmentation(Mat &src, Mat &dst, int thresh, string filename)
{
    Mat srcBlur = src.clone();

    blur(srcBlur,srcBlur,Size(...,...));
    Mat ch[3];
    split(srcBlur,ch);
    subtract(ch[2],ch[1],dst);
    imshow(filename+" segmented", dst);
    threshold(dst,dst,thresh,255,THRESH_BINARY);
    imshow(filename+" binary", dst);
}

void select(Mat &src, Mat &dst, Mat &mask)
{
    dst = src.clone();
    int rows = dst.rows;
    int cols = dst.cols;
    for(int y=0; y<rows; y++)
    {
        uchar *maskPtr = mask.ptr<uchar>(y);
        Vec3b *dstPtr = dst.ptr<Vec3b>(y);
        for(int x=0; x<cols; x++)
        {
            if(maskPtr[x] == 0)
```

```
            {
                dstPtr[x][0] = 0;
                dstPtr[x][1] = 0;
                dstPtr[x][2] = 0;
            }
        }
    }
}

int main()
{
    /*original source: https://pixabay.com/en/tomatoes-red-fresh-
      vegetable-food-1476090/ */
    string filename = "tomato.jpg";
    Mat src = imread(filename);
    Mat segmented;

    int thresh = ...;

    imshow(filename, src);
    tomato_segmentation(src,segmented, thresh, filename);

    int k_size= ...;
    Mat element = getStructuringElement( MORPH_RECT,
    Size( 2*k_size + 1, 2*k_size+1 ), Point( k_size, k_size ) );

    //Lakukan teknik closing disini
      ...
      ...

    //seleksi
    Mat outp;
    select(src, outp, segmented);
    imshow(filename+" output", outp);
    waitKey(0);
    return 0;
}
```

Ekspektasi