LAPORAN TUGAS 5
Visi Komputer dan Pengolahan Citra

Disusun oleh :

Nama : Prihantono
NRP : 1122800017
Prodi : S2 - Teknik Elektro
Dosen : Dr. Setiawardhana, S.T., MT.

P

PROGRAM PASCASARJANA TERAPAN
POLITEKNIK ELEKTRONIKA NEGERI SURABAYA
2023/2024

1. Template Matching menggunakan SAD.

- **Kode Program.**

```
#include <opencv2/opencv.hpp>
#include <iostream>

using namespace cv;
using namespace std;

void templateMatchingSAD(Mat &image, Mat &templateImage) {
   int result_cols = image.cols - templateImage.cols + 1;
   int result_rows = image.rows - templateImage.rows + 1;

   Mat result(result_rows, result_cols, CV_32FC1);

   // Matching dan normalisasi
   matchTemplate(image, templateImage, result, TM_SQDIFF_NORMED);
   normalize(result, result, 0, 1, NORM_MINMAX, -1, Mat());

   // Dapatkan lokasi terbaik
   double minVal, maxVal;
   Point minLoc, maxLoc;
   minMaxLoc(result, &minVal, &maxVal, &minLoc, &maxLoc, Mat());

   // Gambar kotak di sekitar area pencocokan
   rectangle(image, minLoc, Point(minLoc.x + templateImage.cols, minLoc.y +
templateImage.rows), Scalar(0, 255, 0), 2);

   // Tampilkan hasil
   imshow("Result", image);
   waitKey(0);
}

int main() {
   // Baca citra dan template
   Mat image = imread("image.jpg");
   Mat templateImage = imread("template.jpg");

   // Ubah citra dan template ke dalam skala keabuan
   cvtColor(image, image, COLOR_BGR2GRAY);
   cvtColor(templateImage, templateImage, COLOR_BGR2GRAY);

   // Panggil fungsi templateMatchingSAD
   templateMatchingSAD(image, templateImage);

   return 0;
}
```

- **Output program :**

- # Template Matching menggunakan SSD

- **Kode Program :**

```
#include <opencv2/opencv.hpp>
#include <iostream>

using namespace cv;
using namespace std;

void templateMatchingSSD(Mat &image, Mat &templateImage) {
    int result_cols = image.cols - templateImage.cols + 1;
    int result_rows = image.rows - templateImage.rows + 1;

    Mat result(result_rows, result_cols, CV_32FC1);

    // Matching dan normalisasi
    matchTemplate(image, templateImage, result, TM_SQDIFF);
    normalize(result, result, 0, 1, NORM_MINMAX, -1, Mat());

    // Dapatkan lokasi terbaik
    double minVal, maxVal;
    Point minLoc, maxLoc;
    minMaxLoc(result, &minVal, &maxVal, &minLoc, &maxLoc, Mat());
```

```
    // Gambar kotak di sekitar area pencocokan
    rectangle(image, minLoc, Point(minLoc.x + templateImage.cols, minLoc.y +
templateImage.rows), Scalar(0, 255, 0), 2);

    // Tampilkan hasil
    imshow("Result", image);
    waitKey(0);
}

int main() {
    // Baca citra dan template
    Mat image = imread("image.jpg");
    Mat templateImage = imread("template.jpg");

    // Ubah citra dan template ke dalam skala keabuan
    cvtColor(image, image, COLOR_BGR2GRAY);
    cvtColor(templateImage, templateImage, COLOR_BGR2GRAY);

    // Panggil fungsi templateMatchingSSD
    templateMatchingSSD(image, templateImage);

    return 0;
}
```

- **Output Program :**

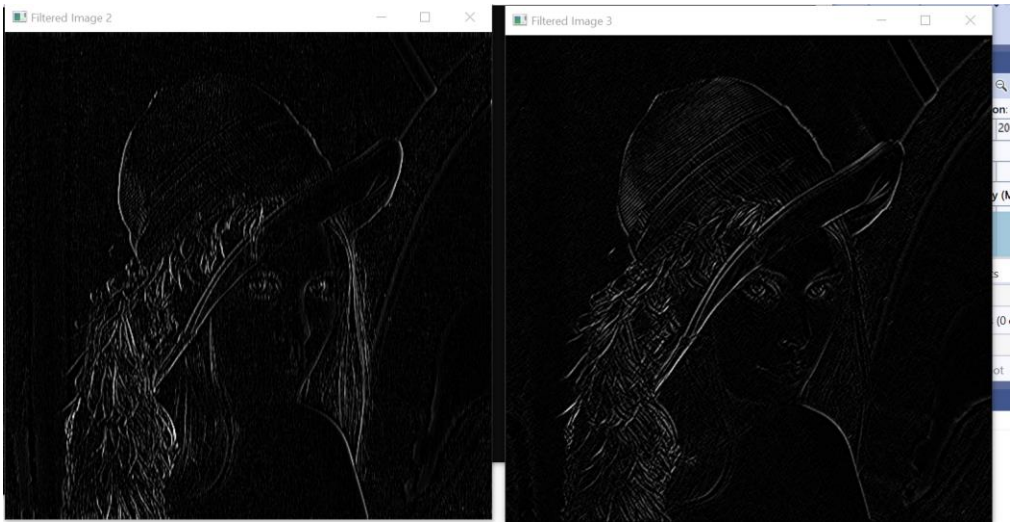
2. Filter Bank.

- **Kode Program :**

```
#include <opencv2/opencv.hpp>
#include <iostream>

using namespace cv;
using namespace std;

// Fungsi untuk mengaplikasikan filter bank ke citra
void applyFilterBank(Mat &image) {
   // Matriks kernel filter bank
   float kernel1[3][3] = {{-1, -1, -1}, {-1, 8, -1}, {-1, -1, -1}};
   float kernel2[3][3] = {{-1, 2, -1}, {-1, 2, -1}, {-1, 2, -1}};
   float kernel3[3][3] = {{-1, -1, 2}, {-1, 2, -1}, {2, -1, -1}};

   // Membuat kernel filter
   Mat filter1 = Mat(3, 3, CV_32F, kernel1);
   Mat filter2 = Mat(3, 3, CV_32F, kernel2);
   Mat filter3 = Mat(3, 3, CV_32F, kernel3);

   // Mengaplikasikan filter ke citra
   Mat result1, result2, result3;
   filter2D(image, result1, -1, filter1);
   filter2D(image, result2, -1, filter2);
   filter2D(image, result3, -1, filter3);

   // Menampilkan hasil
   imshow("Filtered Image 1", result1);
   imshow("Filtered Image 2", result2);
   imshow("Filtered Image 3", result3);
   waitKey(0);
}

int main() {
   // Baca citra
   Mat image = imread("image.jpg");

   // Periksa apakah citra berhasil dibaca
   if (image.empty()) {
      cerr << "Error: Unable to read the image." << endl;
      return -1;
```

```
    }

    // Ubah citra ke dalam skala keabuan
    cvtColor(image, image, COLOR_BGR2GRAY);

    // Panggil fungsi applyFilterBank
    applyFilterBank(image);

    return 0;
}
```

- **Output program :**

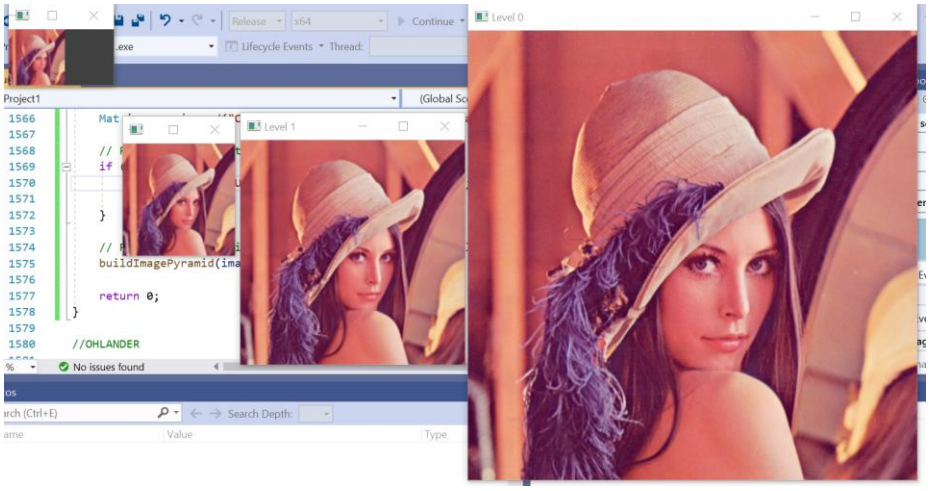


- # Image Pyramid

- **Kode Program :**

```
#include <opencv2/opencv.hpp>
#include <iostream>

using namespace cv;
using namespace std;

// Fungsi untuk membangun image pyramid
void buildImagePyramid(Mat &image, int levels) {
    Mat currentImage = image.clone();

    for (int i = 0; i < levels; ++i) {
        // Tampilkan citra pada setiap level
        imshow("Level " + to_string(i), currentImage);

        // Reduksi citra pada setiap level
        pyrDown(currentImage, currentImage);
    }

    waitKey(0);
}

int main() {
    // Baca citra
    Mat image = imread("image.jpg");

    // Periksa apakah citra berhasil dibaca
    if (image.empty()) {
        cerr << "Error: Unable to read the image." << endl;
        return -1;
    }

    // Panggil fungsi buildImagePyramid dengan jumlah level 4
    buildImagePyramid(image, 4);

    return 0;
}
```

- **Output program :**



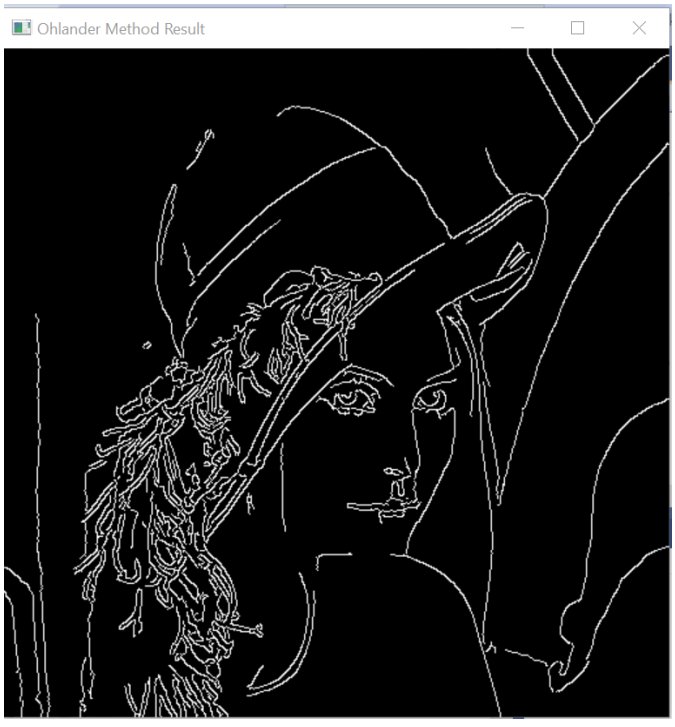
3. Metode Ohlander

- **Kode Program :**

```
#include <opencv2/opencv.hpp>
#include <iostream>

using namespace cv;
using namespace std;

// Fungsi untuk membangun image pyramid dengan metode Ohlander
void buildImagePyramidOhlander(Mat &image, int levels) {
   Mat currentImage = image.clone();

   for (int i = 0; i < levels; ++i) {
      // Tampilkan citra pada setiap level
      imshow("Level " + to_string(i), currentImage);
```

```
        // Reduksi citra pada setiap level menggunakan metode Ohlander
        pyrDown(currentImage, currentImage);

        // Interpolasi untuk menaikkan resolusi
        resize(currentImage, currentImage, image.size(), INTER_LINEAR);
    }

    waitKey(0);
}

int main() {
    // Baca citra
    Mat image = imread("image.jpg");

    // Periksa apakah citra berhasil dibaca
    if (image.empty()) {
        cerr << "Error: Unable to read the image." << endl;
        return -1;
    }

    // Panggil fungsi buildImagePyramidOhlander dengan jumlah level 4
    buildImagePyramidOhlander(image, 4);

    return 0;
}
```

- **Output program :**



4. Transformasi Hough untuk deteksi lingkaran.

- Kode program :

```
#include <opencv2/opencv.hpp>
#include <iostream>

using namespace cv;
using namespace std;

// Fungsi untuk membangun image pyramid dengan metode Ohlander
void buildImagePyramidOhlander(Mat &image, int levels) {
   Mat currentImage = image.clone();

   for (int i = 0; i < levels; ++i) {
      // Tampilkan citra pada setiap level
      imshow("Level " + to_string(i), currentImage);

      // Reduksi citra pada setiap level menggunakan metode Ohlander
      pyrDown(currentImage, currentImage);

      // Interpolasi untuk menaikkan resolusi
      resize(currentImage, currentImage, image.size(), INTER_LINEAR);
   }

   waitKey(0);
}

int main() {
   // Baca citra
   Mat image = imread("image.jpg");

   // Periksa apakah citra berhasil dibaca
   if (image.empty()) {
      cerr << "Error: Unable to read the image." << endl;
      return -1;
   }

   // Panggil fungsi buildImagePyramidOhlander dengan jumlah level 4
   buildImagePyramidOhlander(image, 4);

   return 0;
}
```

Output program :