**COMPUTER VISION AND IMAGE PROCESSING**

**Percobaan 7: Domain Frekuensi**



**Oleh:**

**Luthfi Aminulloh 1020181013**

**TEKNIK ELEKTRO**

**PROGRAM PASCASARJANA S2 TERAPAN**

**POLITEKNIK ELEKTRONIKA NEGERI SURABAYA**

**2019**

# Tujuan Percobaan

1. Mahasiswa mengetahui transformasi citra dari domain waktu diskrit menjadi domain frekuensi diskrit.
2. Mahasiswa memahami bahwa transformasi dari waktu ke frekuensi dapat menunjukkan detail informasi dari sisi lain.

# 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

Transformasi Fourier adalah sebuah cara/teknik analisa sinyal dengan cara mengubah domain sinyal dari domain waktu atau sampel menjadi domain frekuensi. Perubahan domain ini dimaksudkan agar sebuah sinyal dapat dianalisa secara lebih mudah dengan memecah-mecah sinyal yang kompleks/rumit menjadi unsur-unsur sinyal yang lebih kecil atau sederhana.

Transformasi Fourier didefinisikan sebagai sebuah persamaan kontinyu dalam range tertentu sebagai gabungan dari beberapa persamaan sinus. Dasar teori Fourier :

“Ketika sebuah persamaan kontinyu f(t) didefinisikan pada range terbatas, f(t) merupakan gabungan dari gelombang sinus (sebagai gelombang dasar) dengan frekuensi 1/T dan gelombang sinus dengan periode kelipatan bilangan bulat dari frekuensi tersebut”.

Berdasarkan pada teorema Fourier, sebuah informasi berupa sinyal 1-D dimana

memiliki gelombang dasar dengan frekuensi angular m0 = 2n/T, sehingga f(t) menjadi :

Dimana: adalah bilangan tetap (sinyal DC) dan yang selainnya adalah bagian bervibrasi.

Formulasi akhir yang diturunkan dari persamaan diatas menjadi:

Untuk sinyal 2-D, maka persamaan diatas dapat dimodifikasi menjadi:

Dimana: X(p, q) adalah hasil transformasi Fourier, x(k, l) adalah citra masukan pada piksel ke-k dan ke-l, N dan M adalah dimensi citra dalam baris x kolom.

# Prosedur percobaan

## Discrete Fourier Transform 2-D

|  |
| --- |
| List Program: |
| #include<stdlib.h>  #include<math.h>  #include<string.h>  #include<iostream>  #include<opencv2\opencv.hpp>  #include<opencv2\core\core.hpp>  #include<opencv2\highgui\highgui.hpp>  #include<opencv2\imgproc\imgproc.hpp>  #include<opencv2\ml\ml.hpp>  #include<opencv2\objdetect\objdetect.hpp>  using namespace std;  using namespace cv;  //Praktikum 1 Brightness and Contrast  int main() {  cv::Mat I = cv::imread("../data/OrangeCat.jpg", CV\_LOAD\_IMAGE\_GRAYSCALE);  if (I.empty())  return -1;  cv::Mat padded; //expand input image to optimal size  int m = cv::getOptimalDFTSize(I.rows);  int n = cv::getOptimalDFTSize(I.cols); // on the border add zero values  cv::copyMakeBorder(I, padded, 0, m - I.rows, 0, n - I.cols, 1/\*BORDER\_CONSTANT\*/, cv::Scalar::all(0));  cv::Mat planes[] = { cv::Mat\_<float>(padded), cv::Mat::zeros(padded.size(), CV\_32F) };  cv::Mat complexI;  merge(planes, 2, complexI); // Add to the expanded another plane with zeros  dft(complexI, complexI); // this way the result may fit in the source matrix  // compute the magnitude and switch to logarithmic scale  // => log(1 + sqrt(Re(DFT(I))^2 + Im(DFT(I))^2))  split(complexI, planes); // planes[0] = Re(DFT(I), planes[1] = Im(DFT(I))  magnitude(planes[0], planes[1], planes[0]);// planes[0] = magnitude  cv::Mat magI = planes[0];  magI += cv::Scalar::all(1); // switch to logarithmic scale  log(magI, magI);  // crop the spectrum, if it has an odd number of rows or columns  magI = magI(cv::Rect(0, 0, magI.cols & -2, magI.rows & -2));  // rearrange the quadrants of Fourier image so that the origin is at the image center  int cx = magI.cols / 2;  int cy = magI.rows / 2;  cv::Mat q0(magI, cv::Rect(0, 0, cx, cy)); // Top-Left - Create a ROI per quadrant  cv::Mat q1(magI, cv::Rect(cx, 0, cx, cy)); // Top-Right  cv::Mat q2(magI, cv::Rect(0, cy, cx, cy)); // Bottom-Left  cv::Mat q3(magI, cv::Rect(cx, cy, cx, cy)); // Bottom-Right    cv::Mat tmp; // swap quadrants (Top-Left with Bottom-Right)  q0.copyTo(tmp);  q3.copyTo(q0);  tmp.copyTo(q3);  q1.copyTo(tmp); // swap quadrant (top-right with Bottom-left)  q2.copyTo(q1);  tmp.copyTo(q2);  normalize(magI, magI, 0, 1, CV\_MINMAX);  imshow("Input Image", I);  imshow("spectrum magnitude", magI);  waitKey();  return 0;  } |

Hasil Percobaan:

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |