**COMPUTER VISION AND IMAGE PROCESSING**

**Percobaan 4: Point Operators.**



**Oleh:**

**Luthfi Aminulloh 1020181013**

**TEKNIK ELEKTRO**

**PROGRAM PASCASARJANA S2 TERAPAN**

**POLITEKNIK ELEKTRONIKA NEGERI SURABAYA**

**2019**

# Tujuan Percobaan

1. Mahasiswa mengetahui cara dan dapat merekayasa nilai setiap piksel dari gambar untuk proses penambahan nilai brightness dan nilai contrast
2. Mahasiswa mengetahui cara dan dapat mentransformasikan warna dari RGB ke grayscale
3. Mahasiswa mengetahui cara dan dapat menggabungkan dua buah gambar menjadi sebuah gambar lain
4. Mahasiswa dapat merekayasa tingkat gelap-terang gambar menggunakan teknik histogram dan histogram equalization

# 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

## Penyesuaian Brightness dan Contrast

Dua proses titik yang umum digunakan adalah perkalian dan penjumlahan dengan konstanta:

g (x) = αf (x) + β

Parameter α> 0 dan β sering disebut gain dan parameter bias; kadang-kadang parameter ini dikatakan mengontrol contrast dan brightness masing-masing. Kita dapat menganggap f(x) sebagai piksel gambar sumber dan g(x) sebagai piksel gambar output. Kemudian, lebih nyaman kita dapat menulis ekspresi sebagai:

g (i, j) = α⋅f (i, j) + β

di mana i dan j menunjukkan bahwa piksel terletak di baris ke-i dan kolom ke-j.

|  |
| --- |
|  |
| Gambar . Pengaturan brightness dan contrast pada gambar |

## Transformasi Warna

Transformasi dalam ruang RGB seperti menambahkan / menghapus saluran alfa, membalik urutan saluran, konversi ke / dari warna RGB 16-bit (R5: G6: B5 atau R5: G5:B5), serta konversi ke/dari grayscale menggunakan:

RGB[A] ke Gray: Y ← 0,299⋅R + 0,587⋅G + 0,114⋅B

dan

Gray ke RGB[A]: R ← Y, G ← Y, B ← Y, A ← max (ChannelRange)

Konversi dari gambar RGB ke abu-abu dilakukan dengan:

cvtColor (src, bwsrc, cv :: COLOR\_RGB2GRAY);

|  |
| --- |
|  |
| Contoh konversi warna ke grayscale. |

Untuk konversi lainnya, silakan kunjungi situs:

https://docs.opencv.org/3.4.2/de/d25/imgproc\_color\_conversions.html

## Compositing

Operator dyadic (dua-input) yang menarik adalah operator campuran linear yang diformulasikan dalam bentuk:

g (x) = (1 − α).f0(x) + α.f1(x)

Dengan memvariasikan α dari 0 → 1 operator ini dapat digunakan untuk melakukan pembubaran temporal antara dua gambar atau video, seperti yang terlihat dalam tayangan slide dan produksi film.

|  |
| --- |
|  |
| Contoh proses kompositing / blending dua buah gambar |

## Histogram

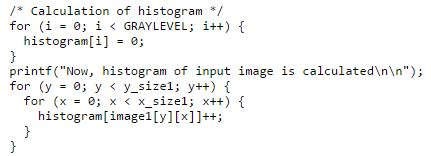
Histogram secara matematis adalah representasi secara grafis dari distribusi data numerik. Dapat berupa perkiraan dari distribusi data sebuah variabel kontinyu (variabel kuantitatif). Histogram digambarkan dengan menggunakan grafik “bar” yang merupakan “bin” dari batas nilai-nilai tertentu (interval) yang tidak overlap/tumpang tindih.

|  |
| --- |
|  |
| Visualisasi histogram dalam matematika |

Histogram citra adalah salah satu tipe histogram yang bertindak sebagai representasi grafis dari distribusi warna dalam sebuah citra digital. Histogram citra menggambarkan jumlah dari piksel setiap nilai warna. Dapat digunakan untuk menentukan distribusi warna dari citra secara keseluruhan.

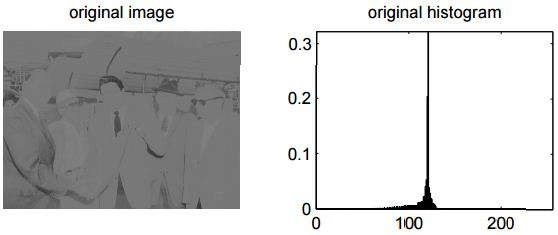
|  |
| --- |
|  |
| Hasil histogram RGB sebuah gambar berwarna |

Gambar 5

Contoh kode untuk kalkulasi histogram dari sebuah citra menggunakan C:

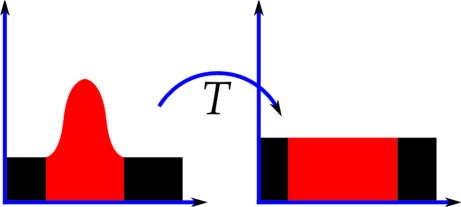
## Histogram Equalization

Histogram equalization adalah sebuah teknik untuk mengatur intensitas citra untuk memperbaiki kontras. Sebuah gambar yang dinyatakan dengan f memiliki ukuran matriks M x N dari nilai intensitas piksel dalam bilangan bulat antara 0 hingga L-1 (misal L = 256 atau 8 bit intensitas atau grayscale).



Gambar 6 Grafik histogram

Metode ini biasanya digunakan untuk menambah kontras global dari gambar, khususnya ketika citra memiliki nilai-nilai intensitas yang berdekatan. Dengan pengaturan ini, diharapkan intensitas citra dapat lebih terdistribusi merata histogramnya.



Gambar 7 Proses equalisasi histogram, meratakan distribusi warna/intensitas

Jika p menyatakan histogram ternormalisasi dari citra f dengan a sebuah “bin” untuk setiap intensitasnya, maka



Gambar yang dihistogram ekualisasi g didefinisikan sebagai:



Untuk memapping balik nilai-nilai pada citra, maka transformasi simpel berikut diperlukan



Berikut ini adalah contoh penerapan histogram equalization pada citra:



Gambar 8 Proses histogram equalisasi gambar

# Prosedur percobaan

## Brightness and Contrast

|  |
| --- |
| List Program: |
| double alpha; // Simple contrast control  int beta; // Simple brightness control  int main()  {  /// Read image given by user  Mat image = imread("cat.jpg"); // nama gambar silakan diganti sesuai gambar yang anda gunakan  Mat new\_image = Mat::zeros(image.size(), image.type());  /// Initialize values  cout << " Basic Linear Transforms " << endl;  cout << "-------------------------" << endl;  cout << "\* Enter the alpha value [1.0-3.0]: ";  cin >> alpha;  cout << "\* Enter the beta value [0-100]: ";  cin >> beta;  /// Do the operation new\_image(i, j) = alpha\*image(i, j) + beta;  for( int y = 0; y < image.rows; y++ )  { for (int x = 0; x < image.cols; x++)  {  for (int c = 0; c < 3; c++)  {  new\_image.at<Vec3b>(y, x)[c] = saturate\_cast<uchar>(alpha\*(image.at<Vec3b>(y, x)[c])+beta);  }  }  }  /// Create Windows namedWindow("Original Image", 1);  namedWindow("New Image", 1);  /// Show stuff  imshow("Original Image", image);  imshow("New Image", new\_image);  /// Wait until user press some key  waitKey();  return 0;  } |

Hasil:

|  |
| --- |
|  |
|  |

**Langkah-langkah percobaan:**

* Masukan nilai alpha berturut-turut 1.0, 1.5, 2.0, dan 3.0 dimana nilai beta = 0! Amati apakah yang terjadi dengan gambar hasilnya!

|  |  |  |  |
| --- | --- | --- | --- |
| 1.0 | 1.5 | 2.0 | 3.0 |
|  |  |  |  |

* Masukan nilai beta berturut-turut 0, 50, dan 100 dimana nilai alpha = 1.0! Amati apakah yang terjadi dengan gambar hasilnya!

|  |  |  |
| --- | --- | --- |
| 0 | 50 | 100 |
|  |  |  |

* Masukkan kombinasi dari nilai alpha dan beta sesuai keinginan anda! Amati apakah yang terjadi pada gambar hasilnya! Jelaskan fenomena yang terjadi!

|  |  |
| --- | --- |
| Alpha = 2.5 || Beta = 25 | Alpha =1.25 || Beta = 85 |
|  |  |

## Color transformation

|  |
| --- |
| Listing Program: |
| int main()  {  Mat image;  image = imread("cat.jpg", CV\_LOAD\_IMAGE\_COLOR);  if (!image.data)  {  cout << "Could not open or find the image" <<  std::endl;  return -1;  }  // Create a new matrix to hold the gray image  Mat gray;  // convert RGB image to gray  cvtColor(image, gray, CV\_BGR2GRAY);  namedWindow("Display window", CV\_WINDOW\_AUTOSIZE);  imshow("Display window", image);  namedWindow("Result window", CV\_WINDOW\_AUTOSIZE);  imshow("Result window", gray);  waitKey(0);  return 0;  } |

Hasil:

|  |
| --- |
|  |
|  |

**Langkah-langkah percobaan:**

* Rekayasalah program diatas dengan membuat program konversi warna dari RGB ke grayscale secara piksel per piksel! (Catatan: Anda harus mengakses informasi per piksel untuk dapat melakukan konversi tersebut).
* Ubahlah program diatas untuk transformasi warna dari RGB  XYZ, RGB  YCrCb, RGB  HSV, dan RGB  Lab! (Catatan: Anda boleh menggunakan fungsi jadi dari OpenCV).
* Untuk poin no 2, rekayasalah masing-masing programnya dengan membuat konversi warna secara piksel per piksel! (Catatan: Anda tidak diperbolehkan menggunakan fungsi jadi dari OpenCV, seperti CvtColor()).

## Compositing and matting

|  |
| --- |
| Listing Program: |
| int main(int argc, char\*\* argv)  {  double alpha = 0.5;  double beta;  double input;  Mat src1, src2, dst;  /// Ask the user enter alpha  cout << " Simple Linear Blender " << endl;  cout << "-----------------------" << endl;  cout << "\* Enter alpha [0-1]: ";  cin >> input;  /// We use the alpha provided by the user if it is between 0 and 1  if( input >= 0.0 && input <= 1.0 )  { alpha = input; }  /// Read image ( same size, same type )  src1 = imread("cat.jpg");  src2 = imread("cat.jpg");  if (!src1.data) { printf("Error loading src1 \n"); return -1; }  if (!src2.data) { printf("Error loading src2 \n"); return -1; }  /// Create Windows  namedWindow("Linear Blend", 1);  beta = (1.0 - alpha);  addWeighted(src1, alpha, src2, beta, 0.0, dst);  imshow("Linear Blend", dst);  waitKey(0);  return 0;  } |

Hasil:

|  |
| --- |
|  |
|  |

**Langkah-langkah percobaan:**

* Rekayasalah program diatas dengan membuat program blending/komposit sendiri tanpa menggunakan fungsi addweighted()! Pergunakan dan aplikasikan rumus dasar komposit yang ada pada teori untuk setiap piksel dari masing-masing gambar!

## Histogram

|  |
| --- |
| Listing Program: |
| int main()  {  Mat src, dst;  /// Load image  src = imread("cat.jpg", 1);  if (!src.data)  {  return -1;  }  /// Separate the image in 3 places ( B, G and R )  vector<Mat> bgr\_planes;  split(src, bgr\_planes);  /// Establish the number of bins  int histSize = 256;  /// Set the ranges ( for B,G,R) )  float range[] = { 0, 256 };  const float\* histRange = { range };    bool uniform = true;  bool accumulate = false;  Mat b\_hist, g\_hist, r\_hist;  /// Compute the histograms:  calcHist(&bgr\_planes[0], 1, 0, Mat(), b\_hist, 1, &histSize, &histRange, uniform, accumulate);  calcHist(&bgr\_planes[1], 1, 0, Mat(), g\_hist, 1, &histSize, &histRange, uniform, accumulate);  calcHist(&bgr\_planes[2], 1, 0, Mat(), r\_hist, 1, &histSize, &histRange, uniform, accumulate);  // Draw the histograms for B, G and R  int hist\_w = 512;  int hist\_h = 400;  int bin\_w = cvRound((double)hist\_w / histSize);  Mat histImage(hist\_h, hist\_w, CV\_8UC3, Scalar(0, 0, 0));  /// Normalize the result to [ 0, histImage.rows ]  normalize(b\_hist, b\_hist, 0, histImage.rows, NORM\_MINMAX, -1, Mat());  normalize(g\_hist, g\_hist, 0, histImage.rows, NORM\_MINMAX, -1, Mat());  normalize(r\_hist, r\_hist, 0, histImage.rows, NORM\_MINMAX, -1, Mat());  /// Draw for each channel  for( int i = 1; i < histSize; i++ )  {  line(histImage, Point(bin\_w\*(i - 1), hist\_h - cvRound(b\_hist.at<float>(i - 1))), Point(bin\_w\*(i), hist\_h - cvRound(b\_hist.at<float>(i))),  Scalar(255, 0, 0), 2, 8, 0);  line(histImage, Point(bin\_w\*(i - 1), hist\_h - cvRound(g\_hist.at<float>(i - 1))), Point(bin\_w\*(i), hist\_h - cvRound(g\_hist.at<float>(i))),  Scalar(0, 255, 0), 2, 8, 0);  line(histImage, Point(bin\_w\*(i - 1), hist\_h - cvRound(r\_hist.at<float>(i - 1))), Point(bin\_w\*(i), hist\_h - cvRound(r\_hist.at<float>(i))),  Scalar(0, 0, 255), 2, 8, 0);  }  /// Display  namedWindow("calcHist Demo", CV\_WINDOW\_AUTOSIZE);  imshow("calcHist Demo", histImage);  waitKey(0);  return 0;  } |

Hasil:

|  |
| --- |
|  |
|  |

**Langkah-langkah percobaan:**

* Ubahlah program diatas untuk menggambarkan histogram grayscale!
* Ubahlah program pada poin 1 (histogram grayscale) dengan menggunakan program buatan anda sendiri! (Catatan: tidak diperbolehkan menggunakan fungsi jadi pada OpenCV seperti: calcHist dan normalize).

## Histogram Equalization

|  |
| --- |
| Listing Program: |
| int main()  {  Mat src, dst;  char\* source\_window = "Source image";  char\* equalized\_window = "Equalized Image";  /// Load image  src = imread("cat.jpg", 1);  if (!src.data)  {  cout << "Usage: ./Histogram\_Demo <path\_to\_image>" << endl;  return -1;  }  /// Convert to grayscale  cvtColor(src, src, CV\_BGR2GRAY);  /// Apply Histogram Equalization  equalizeHist(src, dst);  /// Display results  namedWindow(source\_window, CV\_WINDOW\_AUTOSIZE);  namedWindow(equalized\_window, CV\_WINDOW\_AUTOSIZE);    imshow(source\_window, src);  imshow(equalized\_window, dst);  /// Wait until user exits the program  waitKey(0);  return 0;  } |

**Hasil:**

|  |
| --- |
|  |
|  |

**Langkah-langkah percobaan:**

* Terjemahkan isi dari fungsi equalizeHist() menjadi fungsi buatan anda sendiri! (Catatan: tunjukkan proses histogram equalization secara piksel per piksel).
* Implementasikan fungsi yang telah anda buat tersebut (pada poin 1) untuk melakukan proses histogram equalization! Amati hasilnya, apakah sama dengan fungsi jadi equalizeHist()?

# Tugas

Menambahkan informasi histogram RGB pada sebuah tampilan kamera online.

* Buatlah sebuah program untuk menangkap gambar secara online dengan menggunakan kamera/webcam!
* Kemudian, rekayasalah gambar visualisasinya sehingga muncul informasi grafik histogram warna R, G dan B pada bagian bawah frame gambarnya (menyatu dengan gambar)!

|  |
| --- |
| Listing Program: |
| #include "opencv2/imgproc.hpp"  #include "opencv2/imgcodecs.hpp"  #include "opencv2/highgui.hpp"  #include <iostream>  using namespace cv;  using namespace std;  int main() {  // Don't use global variables if they are not needed!  VideoCapture vid(0);  Mat frame;  while (true)  {  // Read frame  vid.read(frame);  /// Separate the image in 3 places ( B, G and R )  vector<Mat> bgr\_planes;  split(frame, bgr\_planes);  /// Establish the number of bins  int histSize = 256;  /// Set the ranges ( for B,G,R) )  float range[] = { 0, 256 };  const float\* histRange = { range };  bool uniform = true;  bool accumulate = false;  Mat b\_hist, g\_hist, r\_hist;  /// Compute the histograms:  calcHist(&bgr\_planes[0], 1, 0, Mat(), b\_hist, 1, &histSize, &histRange, uniform, accumulate);  calcHist(&bgr\_planes[1], 1, 0, Mat(), g\_hist, 1, &histSize, &histRange, uniform, accumulate);  calcHist(&bgr\_planes[2], 1, 0, Mat(), r\_hist, 1, &histSize, &histRange, uniform, accumulate);  // Draw the histograms for B, G and R  int hist\_w = 512;  int hist\_h = 400;  int bin\_w = cvRound((double)hist\_w / histSize);  Mat histImage(hist\_h, hist\_w, CV\_8UC3, Scalar(0, 0, 0));  /// Normalize the result to [ 0, histImage.rows ]  normalize(b\_hist, b\_hist, 0, histImage.rows, NORM\_MINMAX, -1, Mat());  normalize(g\_hist, g\_hist, 0, histImage.rows, NORM\_MINMAX, -1, Mat());  normalize(r\_hist, r\_hist, 0, histImage.rows, NORM\_MINMAX, -1, Mat());  /// Draw for each channel  for (int i = 1; i < histSize; i++)  {  line(histImage, Point(bin\_w\*(i - 1), hist\_h - cvRound(b\_hist.at<float>(i - 1))), Point(bin\_w\*(i), hist\_h - cvRound(b\_hist.at<float>(i))),  Scalar(255, 0, 0), 2, 8, 0);  line(histImage, Point(bin\_w\*(i - 1), hist\_h - cvRound(g\_hist.at<float>(i - 1))), Point(bin\_w\*(i), hist\_h - cvRound(g\_hist.at<float>(i))),  Scalar(0, 255, 0), 2, 8, 0);  line(histImage, Point(bin\_w\*(i - 1), hist\_h - cvRound(r\_hist.at<float>(i - 1))), Point(bin\_w\*(i), hist\_h - cvRound(r\_hist.at<float>(i))),  Scalar(0, 0, 255), 2, 8, 0);  }  Size sz1 = frame.size();  Size sz2 = histImage.size();  Mat im3(sz1.height, sz1.width+sz2.width, CV\_8UC3);  Mat Left(im3, Rect(0, 0, sz1.width, sz1.height));  frame.copyTo(Left);  Mat Right(im3, Rect(sz1.width,0,sz2.width,sz2.height));  histImage.copyTo(Right);  /// Display  imshow("Webcam", im3);  if ((waitKey(30) & 0xFF) == 27) { // for portability  break;  }  }  } |

|  |
| --- |
|  |
|  |