**COMPUTER VISION AND IMAGE PROCESSING**

**Percobaan 2: Pemanfaatan Library Highgui pada OpenCV.**



**Oleh:**

**Luthfi Aminulloh 1020181013**

**TEKNIK ELEKTRO**

**PROGRAM PASCASARJANA S2 TERAPAN**

**POLITEKNIK ELEKTRONIKA NEGERI SURABAYA**

**2019**

# Tujuan Percobaan

1. Mahasiswa dapat menggunakan library OpenCV untuk menggambar objek dan menulis teks pada kanvas.
2. Mahasiswa dapat melakukan pemotongan gambar berdasarkan region of interest (RoI) yang diinginkan
3. 3. Mahasiswa dapat melakukan kontrol pada citra dengan menggunakan fasilitas obyek kontrol seperti mouse klik dan slider (trackbar).

# 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

Kadang-kadang perlu untuk menampilkan sesuatu ke gambar. Alih-alih menggunakan fungsi yang rumit, mengapa tidak memanggil fungsi sederhana? Berikut ini adalah beberapa fungsi sederhana untuk menghasilkan bentuk-bentuk gambar dan tulisan pada window opencv:

* void circle(image, Point(x,y),int rad, CV\_BGR(b,g,r), int thickness=1)
* void ellipse(image, RotatedRect box, CV\_BGR(b,g,r), int thickness=1)
* void line(image, Point(x,y), Point(x,y), CV\_BGR(b,g,r), int thickness= 1)
* void rectangle(img, Point(x,y), Point(x,y), CV\_BGR(b,g,r), int thickness)

CATATAN: ketebalan negatif akan mengisi persegi panjang

|  |
| --- |
|  |
| Gambar 1. Penggunaan fungsi menggambar dan menulis teks pada window opencv |

ROI adalah bentuk anotasi, sering dikaitkan dengan informasi kategorikal atau kuantitatif (mis., Pengukuran seperti volume atau intensitas rata-rata), diekspresikan sebagai teks atau dalam bentuk terstruktur. Ada tiga cara yang secara fundamental berbeda dalam menyandikan ROI:

* Sebagai bagian integral dari kumpulan data sampel, dengan nilai unik atau masking yang mungkin atau mungkin tidak berada di luar rentang normal dari nilai yang biasanya muncul dan yang menandai sel data individual.
* Sebagai informasi yang terpisah, murni grafis, seperti dengan vektor atau bitmap (rasterisasi) elemen gambar, mungkin dengan beberapa teks polos (tidak terstruktur) dalam format data itu sendiri.
* Sebagai informasi semantik terstruktur terpisah (seperti jenis nilai yang dikodekan) dengan satu set koordinat spasial dan / atau temporal.

|  |
| --- |
|  |
| **Gambar 2.** Contoh region of interest pada gambar |

OpenCV juga memungkinkan Anda menggunakan mouse untuk berinteraksi dengan layar. Fitur ini dari OpenCV 1.0 dan kompatibel dengan objek Mat. Fungsi “mouse\_callback()” ini memungkinkan kita untuk menggambar titik-titik pada gambar.

|  |
| --- |
|  |
| **Gambar 3.** Meletakkan titik-titik hasil click mouse pada area gambar |

Fungsi “createTrackbar()” menciptakan trackbar (slider atau kontrol rentang) dengan nama dan rentang yang ditentukan, menetapkan nilai variabel untuk posisi yang disinkronkan dengan trackbar dan menentukan fungsi callback onChange untuk dipanggil pada perubahan posisi trackbar. Trackbar yang dibuat ditampilkan di winname jendela yang ditentukan. Contoh dibawah ini adalah salah satu penerapan trackbar untuk mengatur nilai contrast dan brightness.

|  |
| --- |
|  |
| **Gambar 4.** Pengaturan nilai contrast dan brightness dengan menggunakan trackbar |

# Prosedur percobaan

## Menggambar dan menulis

|  |
| --- |
| List Program: |
| #include<stdio.h>  #include<stdlib.h>  #include<math.h>  #include<string.h>  #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 cv;  static void help()  {  printf("\nThis program demonstrates OpenCV drawing and text output functions.\n" "Usage:\n" " ./drawing\n");  }  static Scalar randomColor(RNG& rng)  {  int icolor = (unsigned)rng;  return Scalar(icolor & 255, (icolor >> 8) & 255, (icolor >> 16) & 255);  }  int main()  {  help();  char wndname[] = "Drawing Demo";  const int NUMBER = 100;  const int DELAY = 5;  int lineType = LINE\_AA; // change it to LINE\_8 to see non-antialiased graphics  int i, width = 1000, height = 700;  int x1 = -width / 2, x2 = width \* 3 / 2, y1 = -height / 2, y2 = height \* 3 / 2; RNG rng(0xFFFFFFFF);  Mat image = Mat::zeros(height, width, CV\_8UC3); imshow(wndname, image);  waitKey(DELAY);  for (i = 0; i < NUMBER \* 2; i++)  {  Point pt1, pt2;  pt1.x = rng.uniform(x1, x2); pt1.y = rng.uniform(y1, y2); pt2.x = rng.uniform(x1, x2); pt2.y = rng.uniform(y1, y2);  int arrowed = rng.uniform(0, 6); if (arrowed < 3)  line(image, pt1, pt2, randomColor(rng), rng.uniform(1, 10), lineType);  else  arrowedLine(image, pt1, pt2, randomColor(rng), rng.uniform(1, 10), lineType);  imshow(wndname, image); if (waitKey(DELAY) >= 0)  return 0;  }  for (i = 0; i < NUMBER \* 2; i++)  {  Point pt1, pt2;  pt1.x = rng.uniform(x1, x2); pt1.y = rng.uniform(y1, y2); pt2.x = rng.uniform(x1, x2); pt2.y = rng.uniform(y1, y2);  int thickness = rng.uniform(-3, 10); int marker = rng.uniform(0, 10);  int marker\_size = rng.uniform(30, 80);  if (marker > 5)  rectangle(image, pt1, pt2, randomColor(rng), MAX(thickness, -1), lineType);  else  drawMarker(image, pt1, randomColor(rng), marker, marker\_size);  imshow(wndname, image); if (waitKey(DELAY) >= 0)  return 0;  }  for (i = 0; i < NUMBER; i++)  {  Point center;  center.x = rng.uniform(x1, x2); center.y = rng.uniform(y1, y2); Size axes;  axes.width = rng.uniform(0, 200); axes.height = rng.uniform(0, 200); double angle = rng.uniform(0, 180);  ellipse(image, center, axes, angle, angle - 100, angle + 200, randomColor(rng), rng.uniform(-1, 9), lineType);  imshow(wndname, image); if (waitKey(DELAY) >= 0)  return 0;  }  for (i = 0; i< NUMBER; i++)  {  Point pt[2][3];  pt[0][0].x = rng.uniform(x1, x2);  pt[0][0].y = rng.uniform(y1, y2);  pt[0][1].x = rng.uniform(x1, x2);  pt[0][1].y = rng.uniform(y1, y2);  pt[0][2].x = rng.uniform(x1, x2);  pt[0][2].y = rng.uniform(y1, y2);  pt[1][0].x = rng.uniform(x1, x2);  pt[1][0].y = rng.uniform(y1, y2);  pt[1][1].x = rng.uniform(x1, x2);  pt[1][1].y = rng.uniform(y1, y2);  pt[1][2].x = rng.uniform(x1, x2);  pt[1][2].y = rng.uniform(y1, y2);  const Point\* ppt[2] = { pt[0], pt[1] }; int npt[] = { 3, 3 };  polylines(image, ppt, npt, 2, true, randomColor(rng), rng.uniform(1, 10), lineType);  imshow(wndname, image);  if (waitKey(DELAY) >= 0)  return 0;  }  for (i = 0; i< NUMBER; i++)  {  Point pt[2][3];  pt[0][0].x = rng.uniform(x1, x2);  pt[0][0].y = rng.uniform(y1, y2);  pt[0][1].x = rng.uniform(x1, x2);  pt[0][1].y = rng.uniform(y1, y2);  pt[0][2].x = rng.uniform(x1, x2);  pt[0][2].y = rng.uniform(y1, y2);  pt[1][0].x = rng.uniform(x1, x2);  pt[1][0].y = rng.uniform(y1, y2);  pt[1][1].x = rng.uniform(x1, x2);  pt[1][1].y = rng.uniform(y1, y2);  pt[1][2].x = rng.uniform(x1, x2);  pt[1][2].y = rng.uniform(y1, y2);  const Point\* ppt[2] = { pt[0], pt[1] }; int npt[] = { 3, 3 };  fillPoly(image, ppt, npt, 2, randomColor(rng), lineType);  imshow(wndname, image);  if (waitKey(DELAY) >= 0)  return 0;  }  for (i = 0; i < NUMBER; i++)  {  Point center;  center.x = rng.uniform(x1, x2); center.y = rng.uniform(y1, y2);  circle(image, center, rng.uniform(0, 300), randomColor(rng),  rng.uniform(-1, 9), lineType);  imshow(wndname, image);  if (waitKey(DELAY) >= 0)  return 0;  }  for (i = 1; i < NUMBER; i++)  {  Point org;  org.x = rng.uniform(x1, x2); org.y = rng.uniform(y1, y2);  putText(image, "Testing text rendering", org, rng.uniform(0, 8), rng.uniform(0, 100)\*0.05 + 0.1, randomColor(rng), rng.uniform(1, 10), lineType);  imshow(wndname, image);  if (waitKey(DELAY) >= 0)  return 0;  }  Size textsize = getTextSize("OpenCV forever!", FONT\_HERSHEY\_COMPLEX, 3, 5, 0);  Point org((width - textsize.width) / 2, (height - textsize.height) / 2);  Mat image2;  for (i = 0; i < 255; i += 2)  {  image2 = image - Scalar::all(i);  putText(image2, "OpenCV forever!", org, FONT\_HERSHEY\_COMPLEX, 3, Scalar(i, i, 255), 5, lineType);  imshow(wndname, image2); if (waitKey(DELAY) >= 0)  return 0;  }  waitKey();  return 0;  } |

Hasil Percobaan:

|  |
| --- |
|  |
| **Gambar 5.** Percobaan fungsi-fungsi drawing pada opencv |

Langkah-langkah percobaan:

1. Jelaskan apakah yang anda ketahui tentang parameter/argumen (variabel-variabel di dalam kurung) pada beberapa macam fungsi menggambar dan menulis diatas:
   1. **Line**

line(Mat& **img**, Point **pt1**, Point **pt2**, const Scalar& color, int **thickness**=1, int lineType=8, int shift=0).

**Parameters:**

* + img – Image.
  + pt1 – First point of the line segment.
  + pt2 – Second point of the line segment.
  + color – Line color.
  + thickness – Line thickness.
  + lineType – Type of the line:
    - 8 (or omitted) - 8-connected line.
    - 4-connected line.
    - CV\_AA - antialiased line.
  + shift – Number of fractional bits in the point coordinates.
  1. **Arrowedline**

arrowedLine(Mat& img, Point pt1, Point pt2, const Scalar& color, int thickness=1, int line\_type=8, int shift=0, double tipLength=0.1)

**Parameters:**

* + img – Image.
  + pt1 – The point the arrow starts from.
  + pt2 – The point the arrow points to.
  + color – Line color.
  + thickness – Line thickness.
  + line\_type – Type of the line:
    - 8 (or omitted) - 8-connected line.
    - 4-connected line.
    - CV\_AA - antialiased line.
  + shift – Number of fractional bits in the point coordinates.
  + tipLength – The length of the arrow tip in relation to the arrow length
  1. **Rectangle**

**Type 1:** rectangle(Mat& img, Point pt1, Point pt2, const Scalar& color, int thickness=1, int lineType=8, int shift=0)

**Type 2:** rectangle(Mat& img, Rect rec, const Scalar& color, int thickness=1, int lineType=8, int shift=0 )

**Parameters:**

* img – Image.
* pt1 – Vertex of the rectangle.
* pt2 – Vertex of the rectangle opposite to pt1 .
* rec – Alternative specification of the drawn rectangle.
* color – Rectangle color or brightness (grayscale image).
* thickness – Thickness of lines that make up the rectangle. Negative values, like CV\_FILLED , mean that the function has to draw a filled rectangle.
* lineType – Type of the line. See the line() description.
* shift – Number of fractional bits in the point coordinates.
  1. **Drawmarker**

drawMarker (Mat &img, Point position, const Scalar &color, int markerType=MARKER\_CROSS, int markerSize=20, int thickness=1, int line\_type=8)

**Parameters:**

* Img – Image.
* Position – The point where the crosshair is positioned.
* Color – Line color.
* markerType – The specific type of marker you want to use, see MarkerTypes
* thickness – Line thickness.
* line\_type – Type of the line, See LineTypes
* markerSize – The length of the marker axis [default = 20 pixels]
  1. **Ellipse**

**Type 1:** ellipse(Mat& img, Point center, Size axes, double angle, double startAngle, double endAngle, const Scalar& color, int thickness=1, int lineType=8, int shift=0)

**Type 2:** ellipse(Mat& img, const RotatedRect& box, const Scalar& color, int thickness=1, int lineType=8)

**Parameters:**

* img – Image.
* center – Center of the ellipse.
* axes – Half of the size of the ellipse main axes.
* angle – Ellipse rotation angle in degrees.
* startAngle – Starting angle of the elliptic arc in degrees.
* endAngle – Ending angle of the elliptic arc in degrees.
* box – Alternative ellipse representation via RotatedRect or CvBox2D. This means that the function draws an ellipse inscribed in the rotated rectangle.
* color – Ellipse color.
* thickness – Thickness of the ellipse arc outline, if positive. Otherwise, this indicates that a filled ellipse sector is to be drawn.
* lineType – Type of the ellipse boundary. See the line() description.
* shift – Number of fractional bits in the coordinates of the center and values of axes
  1. **Polylines**

**Type 1:** polylines(Mat& img, const Point\*\* pts, const int\* npts, int ncontours, bool isClosed, const Scalar& color, int thickness=1, int lineType=8, int shift=0 )

**Type 2:** polylines(InputOutputArray img, InputArrayOfArrays pts, bool isClosed, const Scalar& color, int thickness=1, int lineType=8, int shift=0 )

**Parameters:**

* img – Image.
* pts – Array of polygonal curves.
* npts – Array of polygon vertex counters.
* ncontours – Number of curves.
* isClosed – Flag indicating whether the drawn polylines are closed or not. If they are closed, the function draws a line from the last vertex of each curve to its first vertex.
* color – Polyline color.
* thickness – Thickness of the polyline edges.
* lineType – Type of the line segments. See the line() description.
* shift – Number of fractional bits in the vertex coordinates.
  1. **Fillpoly**

fillPoly(Mat& img, const Point\*\* pts, const int\* npts, int ncontours, const Scalar& color, int lineType=8, int shift=0, Point offset=Point() )

**Parameters:**

* img – Image.
* pts – Array of polygons where each polygon is represented as an array of points.
* npts – Array of polygon vertex counters.
* ncontours – Number of contours that bind the filled region.
* color – Polygon color.
* lineType – Type of the polygon boundaries. See the line() description.
* shift – Number of fractional bits in the vertex coordinates.
* offset – Optional offset of all points of the contours.
  1. **putText**

putText(Mat& img, const string& text, Point org, int fontFace, double fontScale, Scalar color, int thickness=1, int lineType=8, bool bottomLeftOrigin=false )

**Parameters:**

* img – Image.
* text – Text string to be drawn.
* org – Bottom-left corner of the text string in the image.
* font – CvFont structure initialized using InitFont().
* fontFace – Font type.
* fontScale – Font scale factor that is multiplied by the font-specific base size.
* color – Text color.
* thickness – Thickness of the lines used to draw a text.
* lineType – Line type. See the line for details.
* bottomLeftOrigin – When true, the image data origin is at the bottom-left corner. Otherwise, it is at the top-left corner.

1. Ubah-ubahlah nilai-nilai variabel dalam parameter/argumen tersebut sesuai keinginan anda. Amati apakah yang terjadi?

Fitur drawing yang dikeluarkan oleh setiap fungsi mengikuti nilai parameter input yang diberikan oleh user kecuali nilai yang diberikan dengan nilai random maka dihasilkan fitur acak yang di keluarkan pada jendela luaran.

## Region of Interest

|  |
| --- |
| List Program: |
| #include <Windows.h>  #include <opencv2\highgui\highgui.hpp>  #include <opencv2\imgproc\imgproc.hpp>  #include <opencv2\video\background\_segm.hpp>  #include <opencv2\video\tracking.hpp>  using namespace cv;  using namespace std;  int main(int argc, const char\*\* argv)  {  // Load the image from file Mat LoadedImage;  // Just loaded image Lenna.png from project dir to LoadedImage  Mat LoadedImage = imread("../cat.jpg", IMREAD\_COLOR);  // Show what is in the Mat after load  namedWindow("Step 1 image loaded", WINDOW\_AUTOSIZE);  imshow("Step 1 image loaded", LoadedImage);  waitKey(1000);  // Save the result from LoadedImage to Step1.JPG  imwrite("Step1.JPG", LoadedImage);  // This construct Rectangle Rec start at x=100 y=100, width=200 and heigth=200  Rect Rec(50, 50, 100, 100);  //Draw the rectangle into LoadedImage  //Parameters are (into Mat, Rec describe position where to draw rectangle  // Scalar is Color, 1 is thickness, 8 is line type and 0 shift position  rectangle(LoadedImage, Rec, Scalar(255), 1, 8, 0);  // Show what rectangle  namedWindow("Step 2 draw Rectangle", WINDOW\_AUTOSIZE);  imshow("Step 2 draw Rectangle", LoadedImage);  waitKey(1000);  // Save the result from LoadedImage to Step2.JPG  imwrite("Step2.JPG", LoadedImage);  //Select area described by REC and result write to the Roi  Mat Roi = LoadedImage(Rec);  namedWindow("Step 3 Draw selected Roi", WINDOW\_AUTOSIZE);  imshow("Step 3 Draw selected Roi", Roi);  waitKey(1000);  // Save the result from LoadedImage to Step3.JPG  imwrite("Step3.JPG", Roi);  // Put roi back into source image  // If you want to show the detail and visualize with context of source image  // This rectangle describe target, where you want to  // put your roi into original image  // width and height of where rect must match Roi size  // Let put our Roi into origin  Rect WhereRec(0, 0, Roi.cols, Roi.rows);  // This copy Roi Image into loaded on position Where rec  Roi.copyTo(LoadedImage(WhereRec));  namedWindow("Step 4 Final result", WINDOW\_AUTOSIZE);  imshow("Step 4 Final result", LoadedImage);  waitKey(1000);  // Same the result from LoadedImage to Step4.JPG  imwrite("Step4.JPG", LoadedImage);  } |

Hasil Percobaan:

|  |
| --- |
|  |
| **Gambar 6.** Percobaan menggunakan fungsi RoI |

Langkah-langkah percobaan:

1. Ubah-ubahlah ukuran dari region of interest (ROI) diatas menjadi ukuran yang anda inginkan (syarat tidak boleh melebihi ukuran gambar)!

|  |
| --- |
|  |
| **Gambar 7.** Hasil perubahan nilai frame dari RoI |

2. Jelaskan apa yang anda ketahui tentang “Roi.copyTo(LoadedImage(WhereRec));”!

Roi.CopyTo adalah fungsi untuk menampilkan gambar dari Region of Interest pada Window yang telah ditampilkan dengan letak koordinat (x,y) adalah sesuai dari masukan user. Pada fungsi WhereRec(0, 0, Roi.cols, Roi.rows);

## Mouse Events

|  |
| --- |
| List Program: |
| #include <opencv2\highgui\highgui.hpp>  #include <iostream>  using namespace std;  using namespace cv;  void CallBackFunc(int event, int x, int y, int flags, void\* userdata)  {  if (event == EVENT\_LBUTTONDOWN)  {  cout << "Left button of the mouse is clicked - position (" << x << ", " << y << ")" << endl;  }  else if (event == EVENT\_RBUTTONDOWN)  {  cout << "Right button of the mouse is clicked - position (" << x << ", " << y << ")" << endl;  }  else if (event == EVENT\_MBUTTONDOWN)  {  cout << "Middle button of the mouse is clicked - position (" << x << ", " << y << ")" << endl;  }  else if (event == EVENT\_MOUSEMOVE)  {  cout << "Mouse move over the window - position (" << x << ", " << y << ")" << endl;  }  }  int main(int argc, char\*\* argv)  {  // Read image from file  Mat img = imread("../cat.jpg");  // gunakan gambar yang anda inginkan (jpg, bmp, png)  //if fail to read the image  if ( img.empty() )  {  cout << "Error loading the image" << endl;  return -1;  }  //Create a window  namedWindow("Bismillah", 1);  //set the callback function for any mouse event  setMouseCallback("Bismillah", CallBackFunc, NULL);  //show the image  imshow("Bismillah", img);  // Wait until user press some key  waitKey(0);  return 0;  } |

Hasil Percobaan:

|  |
| --- |
|  |
| **Gambar 8.** Menampilkan posisi dari mouse event |

Langkah-langkah percobaan:

1. Apakah yang anda ketahui tentang:

a. EVENT\_LBUTTONDOWN

Mengindikasikan bahwa tombol mouse sebelah kiri sedang di tekan.

b. EVENT\_RBUTTONDOWN

Mengindikasikan bahwa tombol mouse sebelah kanan sedang di tekan.

c. EVENT\_MBUTTONDOWN

Mengindikasikan bahwa scroll pada mouse sedang di tekan.

d. EVENT\_MMOUSEMOVE

Menunjukkan bahwa penunjuk (posisi) pada mouse telah pindah pada jendela.

2. Modifikasilah program diatas untuk mendapatkan nilai RGB pada posisi yang di-klik dengan mouse!

|  |
| --- |
| void mouseEvent(int evt, int x, int y, int flags, void\* param)  {  Mat\* rgb = (Mat\*)param;  if (evt == CV\_EVENT\_LBUTTONDOWN)  {  printf("Nilai pixel pada koordinat (%d,%d): B=%d, G=%d, R=%d\n",x, y,  (int)(\*rgb).at<Vec3b>(y, x)[0],  (int)(\*rgb).at<Vec3b>(y, x)[1],  (int)(\*rgb).at<Vec3b>(y, x)[2]);  }  } |

|  |
| --- |
|  |
| **Gambar 9.** Mengambil nilai RGB pada pixel tertentu dengan mouse event |

## Trackbar (slide) control

|  |
| --- |
| List Program: |
| #include<opencv2\imgproc\imgproc.hpp>  #include<opencv2\highgui\highgui.hpp>  #include<opencv2\imgcodecs.hpp>  using namespace cv;  /// Global variables  int threshold\_value = 0;  int threshold\_type = 3;  int const max\_value = 255;  int const max\_type = 4;  int const max\_BINARY\_value = 255;  Mat src, src\_gray, dst;  const char\* window\_name = "Threshold Demo";  const char\* trackbar\_type = "Type: \n 0: Binary \n 1: Binary Inverted \n 2: Truncate \n 3: To Zero \n 4: To Zero Inverted";  const char\* trackbar\_value = "Value";  /// Function headers  void Threshold\_Demo(int, void\*);  /\*\*  \* @function main  \*/  int main(int argc, char\*\* argv)  {  //! [load]  String imageName("cat.jpg"); // by default  if (argc > 1)  {  imageName = argv[1];  }  src = imread(imageName, IMREAD\_COLOR); // Load an image  if (src.empty())  {  return -1;  }  cvtColor(src, src\_gray, COLOR\_BGR2GRAY); // Convert the image to Gray  //! [load]  //! [window]  // Create a window to display results  namedWindow(window\_name, WINDOW\_AUTOSIZE);    //! [window]  //! [trackbar]  createTrackbar(trackbar\_type, window\_name, &threshold\_type, max\_type, Threshold\_Demo);  // Create Trackbar to choose type of Threshold  createTrackbar(trackbar\_value, window\_name, &threshold\_value,  max\_value, Threshold\_Demo);  // Create Trackbar to choose Threshold value  //! [trackbar]  Threshold\_Demo(0, 0);  // Call the function to initiali  /// Wait until user finishes program  for (;;)  {  char c = (char)waitKey(20);  if (c == 27)  {  break;  }  }  }  void Threshold\_Demo(int, void\*)  {  /\* 0: Binary  1: Binary Inverted  2: Threshold Truncated 3: Threshold to Zero  4: Threshold to Zero Inverted  \*/  threshold(src\_gray, dst, threshold\_value, max\_BINARY\_value, threshold\_type);  imshow(window\_name, dst);  } |

Hasil Percobaan:

|  |
| --- |
|  |
| **Gambar 10.** Percobaan perubahan nilai pada gambar dengan manipulasi trackbar |

Langkah-langkah percobaan:

1. Jelaskanlah maksud dan cara kerja dari program diatas!

Pada program trackbar slide control dibuat fungsi trackbar dengan 5 pilihan yaitu Binary, binary inverted, Truncate, To zero, To Zero inverted yang mana trackbar memiliki nilai mulai dari 0 sampai 255 dengan nilai tersebut sebagai variabel masukan dari type trackbar yang digunakan sehingga dengan melakukan penggeseran pada trackbar akan mempengaruhi fungsi yang bekerja pada gambar yang diolah sesuai dengan type trackbar yang di pilih.

2. Jelaskan secara detail tentang fungsi “createTaskbar()” diatas! Apakah maksud dari setiap argumentasi/variabel yang ada dalam tanda kurung pada fungsi tersebut?

createTrackbar(const string& **trackbarname**, const string& **winname**, int\* **value**, int **count**, TrackbarCallback **onChange**=0, void\* **userdata**=0)

**Parameters:**

* trackbarname – Name of the created trackbar.
* winname – Name of the window that will be used as a parent of the created trackbar.
* value – Optional pointer to an integer variable whose value reflects the position of the slider. Upon creation, the slider position is defined by this variable.
* count – Maximal position of the slider. The minimal position is always 0.
* onChange – Pointer to the function to be called every time the slider changes position. This function should be prototyped as void Foo(int,void\*); , where the first parameter is the trackbar position and the second parameter is the user data (see the next parameter). If the callback is the NULL pointer, no callbacks are called, but only value is updated.
* userdata – User data that is passed as is to the callback. It can be used to handle trackbar events without using global variables.

## Moving around video with slider

|  |
| --- |
| List Program: |
| #include <opencv2\highgui\highgui.hpp>  #include <opencv2\opencv.hpp>  int g\_slider\_position = 0;  CvCapture\* g\_capture = NULL;  void onTrackbarSlide(int pos)  {  cvSetCaptureProperty(  g\_capture,  CV\_CAP\_PROP\_POS\_FRAMES,  pos  );  }  int main( int argc, char\*\* argv )  {  cvNamedWindow( "AVI", CV\_WINDOW\_AUTOSIZE );  g\_capture = cvCreateFileCapture("../Megamind.avi");  int frames = (int) cvGetCaptureProperty(  g\_capture,  CV\_CAP\_PROP\_FRAME\_COUNT  );  if( frames!= 0 ) {  cvCreateTrackbar(  "Position",  "AVI",  &g\_slider\_position,  frames,  onTrackbarSlide  );  }  IplImage\* frame;  while(1)  {  frame = cvQueryFrame( g\_capture );  if( !frame ) break;  cvShowImage( "AVI", frame );  char c = cvWaitKey(25);  if( c == 27 ) break;  }  cvReleaseCapture( &g\_capture );  cvDestroyWindow( "AVI" );  return(0);  } |

Hasil Percobaan:

Langkah-langkah percobaan:

1. Ubah nama “video.avi” dengan nama file video yang anda punyai, kemudian jalankan program!

|  |
| --- |
|  |
| **Gambar 11.** Error versi opencv tidak support dengan program yang dijalankan |

2. Jelaskan fungsi berikut ini beserta dengan parameter yang ada di dalamnya.

a. onTrackbarSlide()

b. cvQueryFrame()

# Tugas

* **Menulis video pada file**

Dari program menampilkan PC camera pada praktikum1, buatlah program untuk menyimpan tampilan tersebut ke dalam bentuk video (\*.avi) dalam rentang waktu beberapa saat setelah program pertama kali dijalankan.

|  |
| --- |
| Listing Program: |
| #include <iostream>  #include<opencv2\opencv.hpp>  using namespace std;  using namespace cv;  int main() {  // Create a VideoCapture object and use camera to capture the video  VideoCapture cap(0);  // Check if camera opened successfully  if (!cap.isOpened())  {  cout << "Error opening video stream" << endl;  return -1;  }  // Default resolution of the frame is obtained.The default resolution is system dependent.  int frame\_width = cap.get(CV\_CAP\_PROP\_FRAME\_WIDTH);  int frame\_height = cap.get(CV\_CAP\_PROP\_FRAME\_HEIGHT);  // Define the codec and create VideoWriter object.The output is stored in 'outcpp.avi' file.  VideoWriter video("out.avi", CV\_FOURCC('M', 'J', 'P', 'G'), 10, Size(frame\_width, frame\_height));  while (1)  {  Mat frame;  // Capture frame-by-frame  cap >> frame;  // If the frame is empty, break immediately  if (frame.empty())  break;  // Write the frame into the file 'outcpp.avi'  video.write(frame);  // Display the resulting frame  imshow("Frame", frame);  // Press ESC on keyboard to exit  char c = (char)waitKey(1);  if (c == 27)  break;  }  // When everything done, release the video capture and write object  cap.release();  video.release();  // Closes all the windows  destroyAllWindows();  return 0;  } |

Hasil:

|  |
| --- |
|  |
| **Gambar 12.** Video yang diambil menggunakan webcam |

|  |
| --- |
|  |
| **Gambar 13.** File hasil penyimpanan video yang telah direkam |