**Problem Set 3  
Pengantar Pengolahan Citra Digital**Oleh Nino Tannio H.– Lab CI IPB

Pengumpulan:

To: nino\_tannio13@apps.ipb.ac.id

Subject: PPCD3\_P1/2\_NIM

Deadline: 27 Desember 2017­

File: rar/zip (file doc dan file program yang sudah dihapus executable filenya)

|  |
| --- |
| Task 1:   * Kembangkan fitur Hough Transform Circle agar dapat menghitung jumlah lingkaran (lakukan pengujian dengan menggunakan 3 gambar yang berbeda) * Kembangkan inputan dengan menggunakan media video dengan memanfaatkan fitur HT Circle * Tambahkan fungsi baru (dan pilihan transform) dengan menggunakan teknik ‘blob’, ‘countour’, dan ‘template matching’ * Buat program baru dengan memanfaatkan teknik Haar Classifier |
| Task 2 (opsional):   * Buat sebuah program deteksi dengan input video(cam) dengan memanfaatkan fitur Hough Transform line untuk menghitung jumlah objek yang melewati garis yang diletakkan di tengah frame. (misalnya koin yang dilempar dari atas ke bawah) lalu munculkan jumlah hitung realtimenya pada layar. |

Nama : Rya Meyvriska

NIM : G64164008

LAB : Praktikum 2

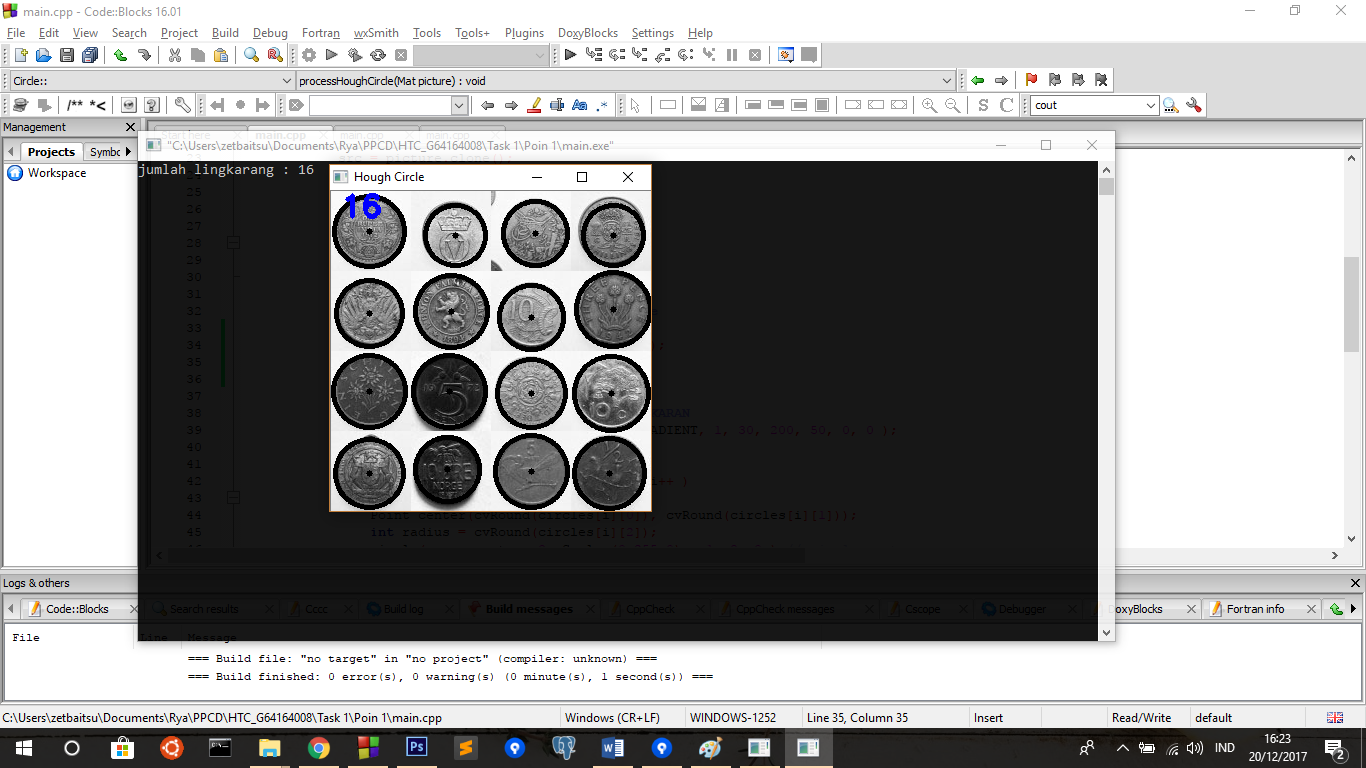
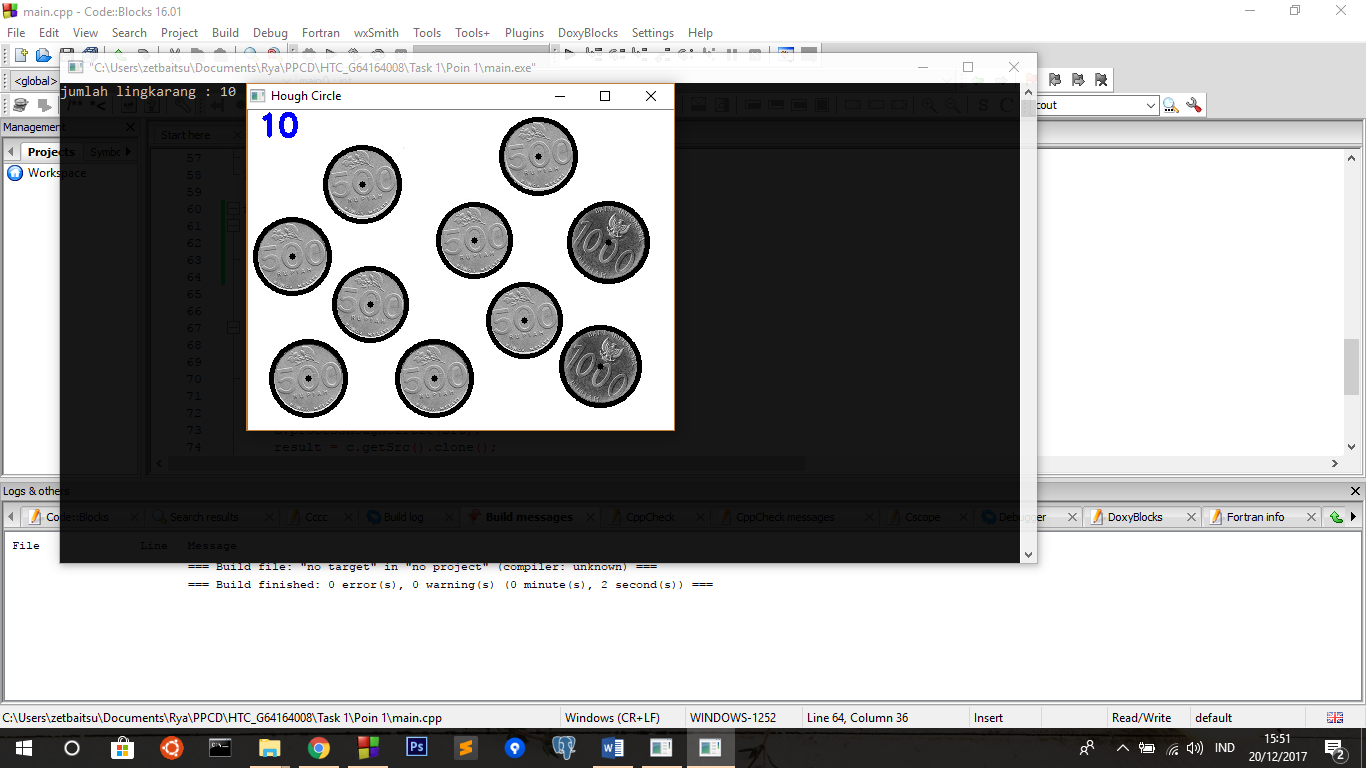
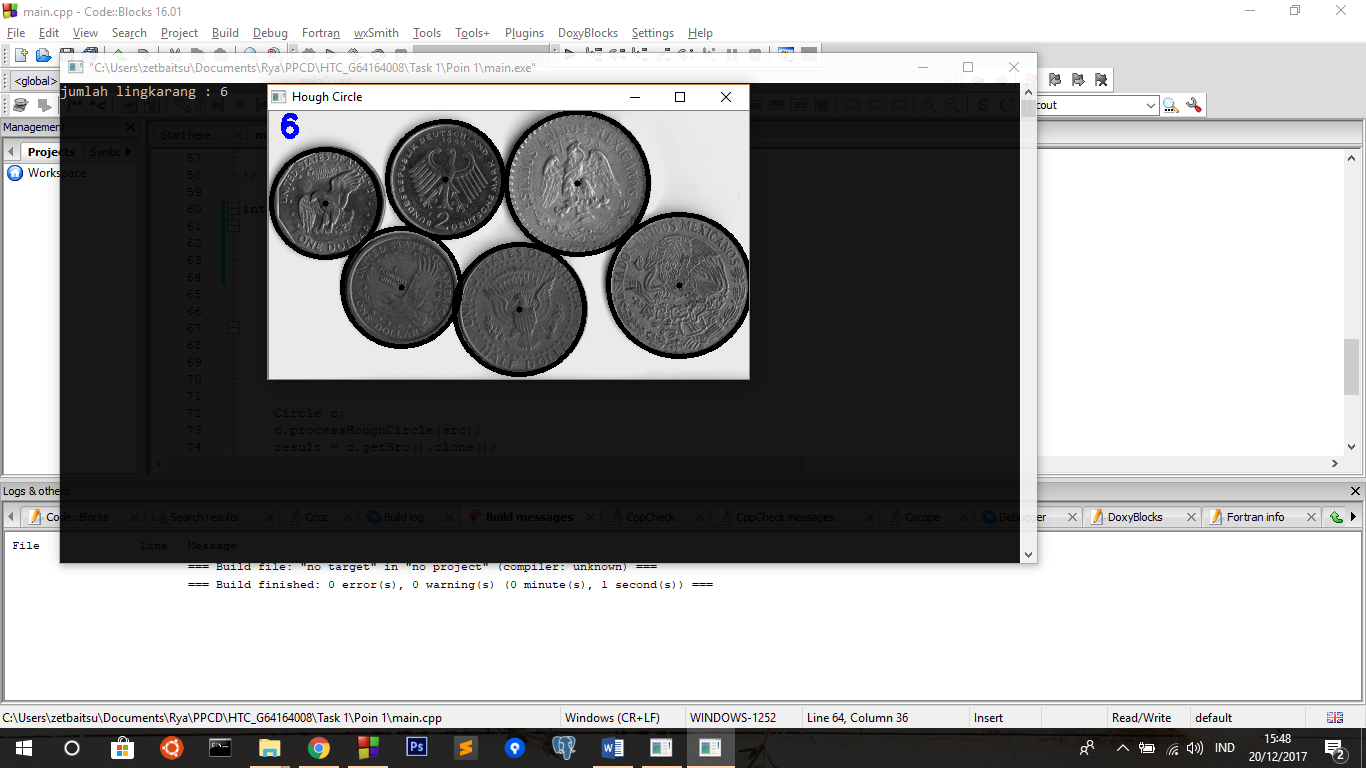
Task 1 – Poin 1

Kembangkan fitur Hough Transform Circle agar dapat menghitung jumlah lingkaran (lakukan pengujian dengan menggunakan 3 gambar yang berbeda)

Main.cpp

|  |
| --- |
| /\*  Rya Meyvriska  G64164008  Task 1 - Poin 1  \*/  #include "opencv2/highgui/highgui.hpp"  #include "opencv2/imgproc/imgproc.hpp"  #include <iostream>  #include <iomanip>  #include <sstream>  using namespace cv;  using namespace std;  #include "contrast.h"  class Circle{  private:  Mat dst,src;  int sum;  public:  void processHoughCircle(Mat picture){  src = picture.clone();  dst = src.clone();  sum = 0;  /// Ubah to Gray  if(src.channels()==3){  cvtColor(src,src,CV\_BGR2GRAY);  }  /// img enhanchment untuk mereduce noise  src = contrast(src);  GaussianBlur(src,dst, Size(3, 3), 2, 2 );  medianBlur(dst,dst,3);  vector<Vec3f> circles;  /// LIBRARY HOUGH UNTUK MENDETEKSI LINGKARAN  HoughCircles( dst, circles, CV\_HOUGH\_GRADIENT, 1, 30, 200, 50, 0, 0 );  /// ALGORITMA MENGGAMBAR LINGKARAN  for( size\_t i = 0; i < circles.size(); i++ )  {  Point center(cvRound(circles[i][0]), cvRound(circles[i][1]));  int radius = cvRound(circles[i][2]);  circle( src, center, 3, Scalar(0,255,0), -1, 8, 0 );// circle center  circle( src, center, radius, Scalar(0,0,255), 3, 8, 0 );// circle outline  }  sum = circles.size();  }  Mat getSrc(){  return src;  }  int getSum(){  return sum;  }  };  int main(){  /\*  ada coin1-coin5.jpg  \*/  Mat src = imread("picture/coin1.jpg", 0);  Mat result;  if(!src.data){  cout<<"file gambar tidak tersedia"<<endl;  return -1;  }  Circle c;  c.processHoughCircle(src);  result = c.getSrc().clone();  cvtColor(result,result,CV\_GRAY2BGR);  ostringstream sum;  sum << c.getSum();  putText(result,sum.str(),Point(10,25),FONT\_HERSHEY\_SIMPLEX,1,Scalar(255,0,0),4);  imshow("Hough Circle", result);  moveWindow("Hough Circle",512,0);  cout<< "jumlah lingkarang : "<<c.getSum()<<endl;  waitKey();  return 0;  } |

Screenshoot:



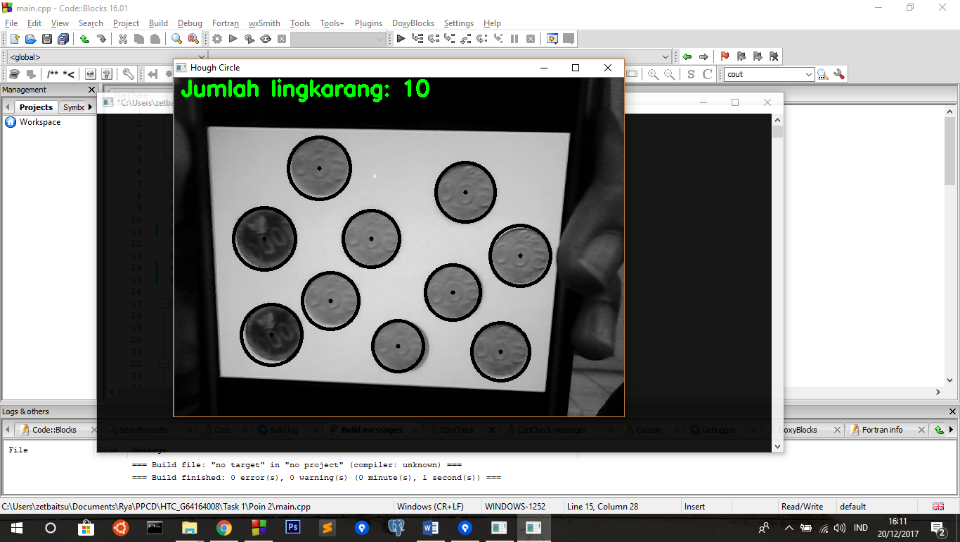
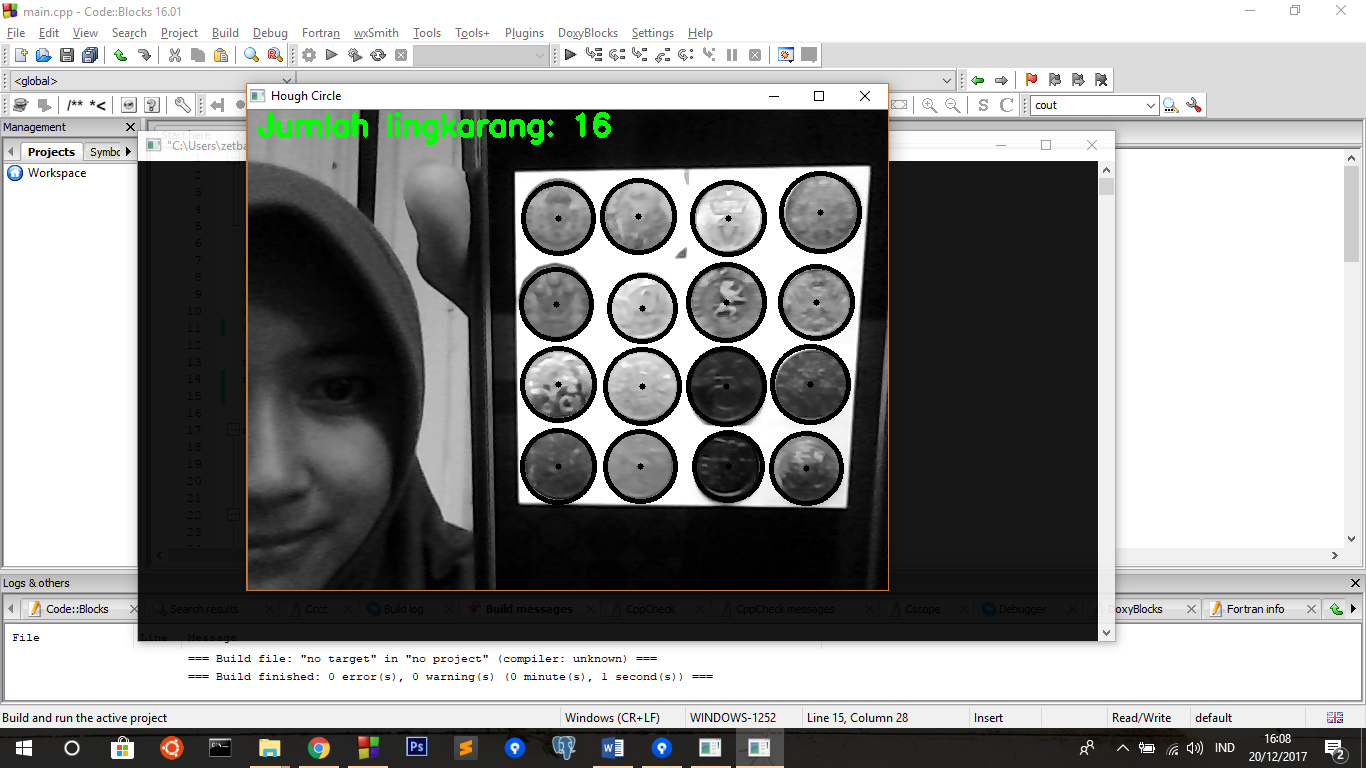
Task 1 – Poin 2

Kembangkan inputan dengan menggunakan media video dengan memanfaatkan fitur HT Circle

Main.cpp

|  |
| --- |
| /\*  Rya Meyvriska  G64164008  Task 1  \*/  #include "opencv2/highgui/highgui.hpp"  #include "opencv2/imgproc/imgproc.hpp"  #include <iomanip>  #include <iostream>  #include <sstream>  using namespace cv;  using namespace std;  #include "../../contrast.h"  class Circle{  private:  Mat dst,src;  int sum;  public:  void processHoughCircle(Mat picture){  src = picture.clone();  dst = src.clone();  sum = 0;  /// Ubah to Gray  if(src.channels()==3){  cvtColor(src,src,CV\_BGR2GRAY);  }  /// img enhanchment untuk mereduce noise  src = contrast(src);  GaussianBlur(src,dst, Size(3, 3), 2, 2 );  medianBlur(dst,dst,3);  vector<Vec3f> circles;  /// LIBRARY HOUGH UNTUK MENDETEKSI LINGKARAN  HoughCircles( dst, circles, CV\_HOUGH\_GRADIENT, 1, 30, 200, 50, 0, 0 );  /// ALGORITMA MENGGAMBAR LINGKARAN  for( size\_t i = 0; i < circles.size(); i++ )  {  Point center(cvRound(circles[i][0]), cvRound(circles[i][1]));  int radius = cvRound(circles[i][2]);  circle( src, center, 3, Scalar(0,255,0), -1, 8, 0 );// circle center  circle( src, center, radius, Scalar(0,0,255), 3, 8, 0 );// circle outline  }  sum = circles.size();  }  Mat getSrc(){  return src;  }  int getSum(){  return sum;  }  };  int main(){  VideoCapture cap(0); // open the default camera  if(!cap.isOpened()) // check if we succeeded  return -1;  Circle c;  while(true)  {  Mat frame,result;  cap >> frame; // get a new frame from camera  flip(frame, frame,1); //agar tidak mirror  c.processHoughCircle(frame);  result = c.getSrc().clone();  cvtColor(result,result,CV\_GRAY2BGR);  ostringstream sum;  sum << c.getSum();  putText(result,"Jumlah lingkarang: " + sum.str(),Point(10,25),FONT\_HERSHEY\_SIMPLEX,1,Scalar(0,255,0),4);  imshow("Hough Circle", result);  if(waitKey(1) >= 0) break;  }  return 0;  } |

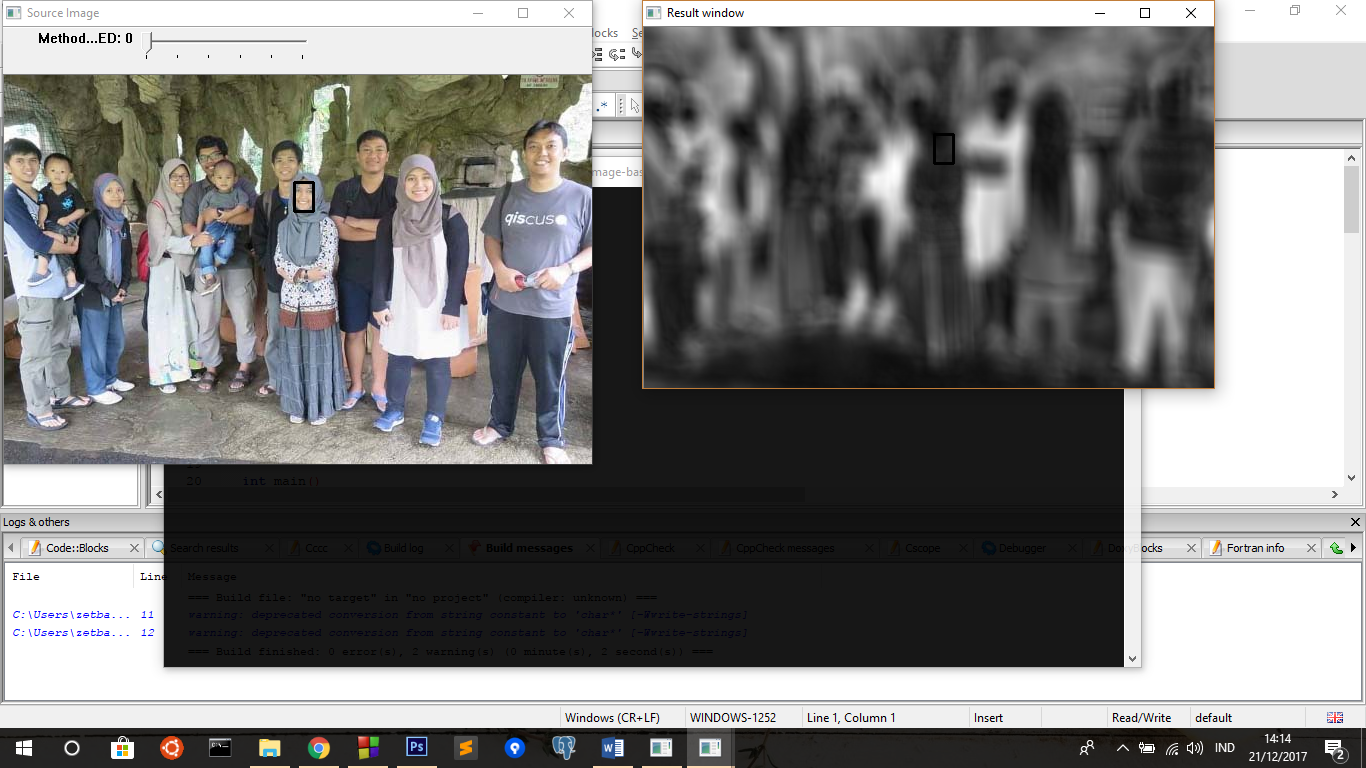
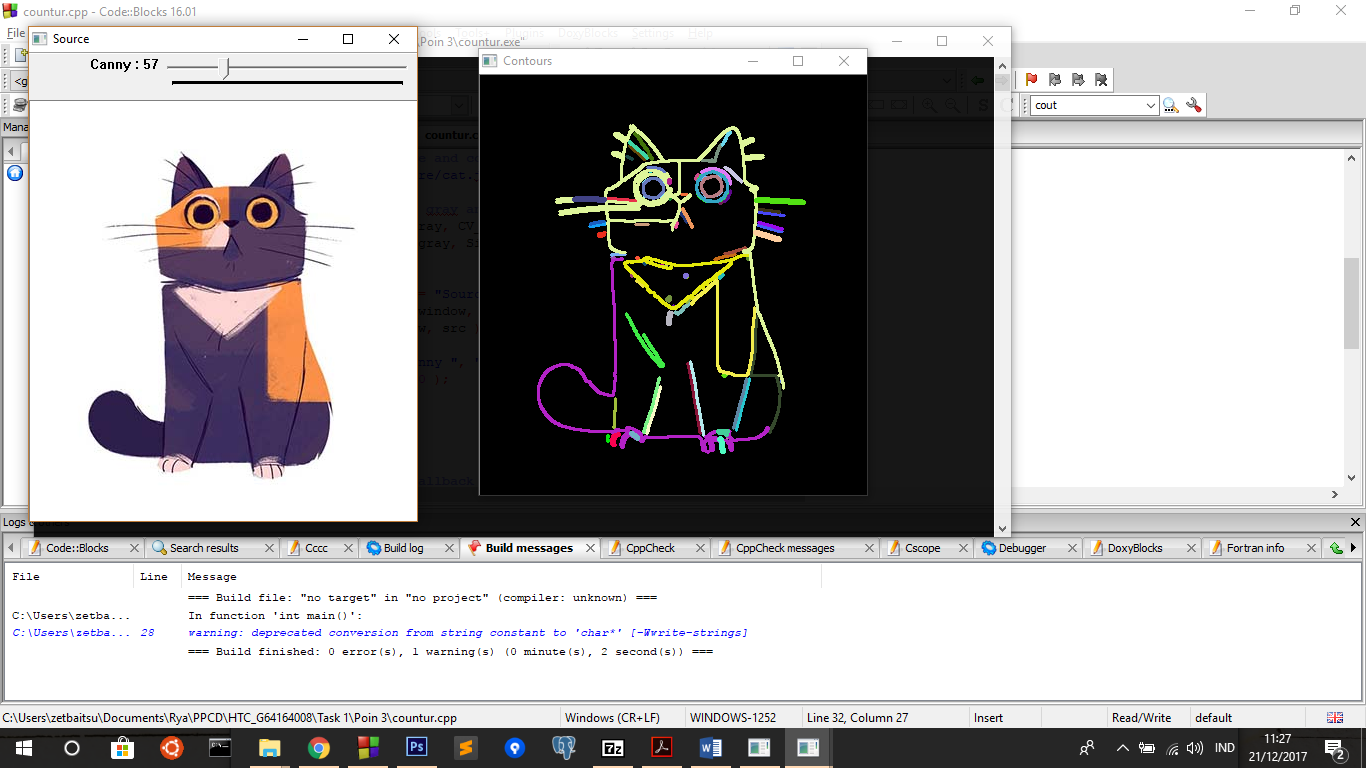
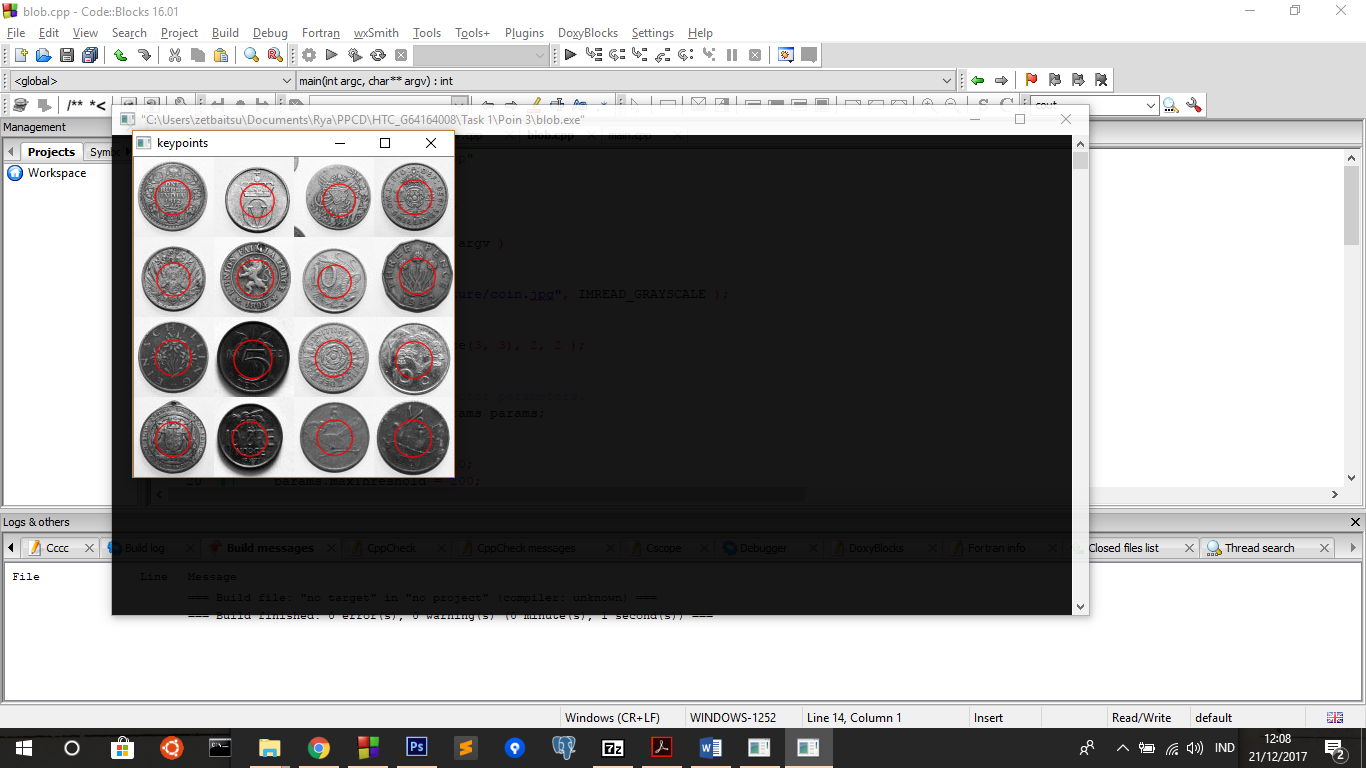
Screenshoot:



Task 1 – Poin 3

Tambahkan fungsi baru (dan pilihan transform) dengan menggunakan teknik ‘blob’, ‘countour’, dan ‘template matching’

Pada poin ini saya bingung, dibuat image base per tekhnik, atau melanjutkan dari video. Saya sudah mencoba bertanya di kelas, namun jawabannya tetap membingungkan saya. Sehingga saya membuat 2 base, image base dan video base. Berikut screenshoot image base:



Gambar Screenshoot Blob Teknik, Countur Teknik, dan Template Method Teknik

\* code image base bisa langsung dilihat di source code terlampir, disini saya menyertakan langsung source code untuk video base

Code pada video base:

Main.cpp

|  |
| --- |
| /\*  Rya Meyvriska  G64164008  Task 1 Poin 2  \*/  #include "opencv2/highgui/highgui.hpp"  #include "opencv2/imgproc/imgproc.hpp"  #include <iomanip>  #include <iostream>  #include <sstream>  using namespace cv;  using namespace std;  #include "../../../contrast.h"  #include "header/blob.cpp"  #include "header/countour.cpp"  #include "header/template\_matching.cpp"  class Circle{  private:  Mat dst,src;  int sum;  public:  void processHoughCircle(Mat picture){  src = picture.clone();  dst = src.clone();  sum = 0;  /// Ubah to Gray  if(src.channels()==3){  cvtColor(src,src,CV\_BGR2GRAY);  }  /// img enhanchment untuk mereduce noise  src = contrast(src);  GaussianBlur(src,dst, Size(3, 3), 2, 2 );  medianBlur(dst,dst,3);  vector<Vec3f> circles;  /// LIBRARY HOUGH UNTUK MENDETEKSI LINGKARAN  HoughCircles( dst, circles, CV\_HOUGH\_GRADIENT, 1, 30, 200, 50, 0, 0 );  /// ALGORITMA MENGGAMBAR LINGKARAN  for( size\_t i = 0; i < circles.size(); i++ )  {  Point center(cvRound(circles[i][0]), cvRound(circles[i][1]));  int radius = cvRound(circles[i][2]);  circle( src, center, 3, Scalar(0,255,0), -1, 8, 0 );// circle center  circle( src, center, radius, Scalar(0,0,255), 3, 8, 0 );// circle outline  }  sum = circles.size();  }  Mat getSrc(){  return src;  }  int getSum(){  return sum;  }  };  Mat addText(Mat src, String str, int x, int y){  putText(src,str,Point(x,y),FONT\_HERSHEY\_COMPLEX\_SMALL,1,Scalar(255,255,255),1);  return src;  }  void printDescription(int type){  Mat des = Mat::zeros(200,380,type);  des = addText(des,"Tekan:",10,25);  des = addText(des,"'b' untuk teknik blob",10,50);  des = addText(des,"'c' untuk teknik countour",10,75);  des = addText(des,"'t' untuk template maching",10,100);  des = addText(des,"'r' untuk original",10,125);  des = addText(des,"'q' untuk keluar",10,150);  imshow("Description",des);  moveWindow("Description",650,0);  }  int main(){  VideoCapture cap(0); // open the default camera  if(!cap.isOpened()) // check if we succeeded  return -1;  Circle c;  bool blob = false;  bool countour = false;  bool templateMatching = false;  while(true)  {  Mat frame,result;  cap >> frame; // get a new frame from camera  flip(frame, frame,1); //agar tidak mirror  c.processHoughCircle(frame);  result = c.getSrc().clone();  if(blob){  result = blobTekhnik(result);  }  if(countour){  result = countourTekhnik(result);  }  if(templateMatching){  Mat temp = imread("template.jpg");  cvtColor(temp,temp,CV\_BGR2GRAY);  result = templateMatchingTekhnik(result,temp);  }  if(result.channels()==1)  cvtColor(result,result,CV\_GRAY2BGR);  ostringstream sum;  sum << c.getSum();  putText(result,"Jumlah lingkarang: " + sum.str(),Point(10,25),FONT\_HERSHEY\_SIMPLEX,1,Scalar(0,255,0),4);  printDescription(result.type());  imshow("Hough Circle", result);  moveWindow("Hough Circle", 0,0);  int key = waitKey(100);  if(key=='b'){  blob = true;  countour = false;  templateMatching = false;  }else if(key=='c'){  blob = false;  countour = true;  templateMatching = false;  }else if(key=='t'){  blob = false;  countour = false;  templateMatching = true;  }else if(key=='r'){  blob = false;  countour = false;  templateMatching = false;  }else if(key=='q'){  cout<<"terimakasih"<<endl;  break;  }  }  return 0;  } |

Header blob.cpp

|  |
| --- |
| #include "opencv2/opencv.hpp"  Mat blobTekhnik(Mat src){  Mat im = src.clone();  im = contrast(im);  GaussianBlur(im,im, Size(3, 3), 2, 2 );  medianBlur(im,im,3);  // Setup SimpleBlobDetector parameters.  SimpleBlobDetector::Params params;  // Change thresholds  params.minThreshold = 10;  params.maxThreshold = 200;  // Filter by Area.  params.filterByArea = true;  params.minArea = 1500;  // Filter by Circularity  params.filterByCircularity = true;  params.minCircularity = 0.1;  // Filter by Convexity  params.filterByConvexity = true;  params.minConvexity = 0.87;  // Filter by Inertia  params.filterByInertia = true;  params.minInertiaRatio = 0.01;  // Storage for blobs  vector<KeyPoint> keypoints;  // Set up detector with params  SimpleBlobDetector detector(params);  // Detect blobs  detector.detect( im, keypoints);  Mat im\_with\_keypoints;  drawKeypoints( src, keypoints, im\_with\_keypoints, Scalar(0,0,255), DrawMatchesFlags::DRAW\_RICH\_KEYPOINTS );  // return blobs  return im\_with\_keypoints;  } |

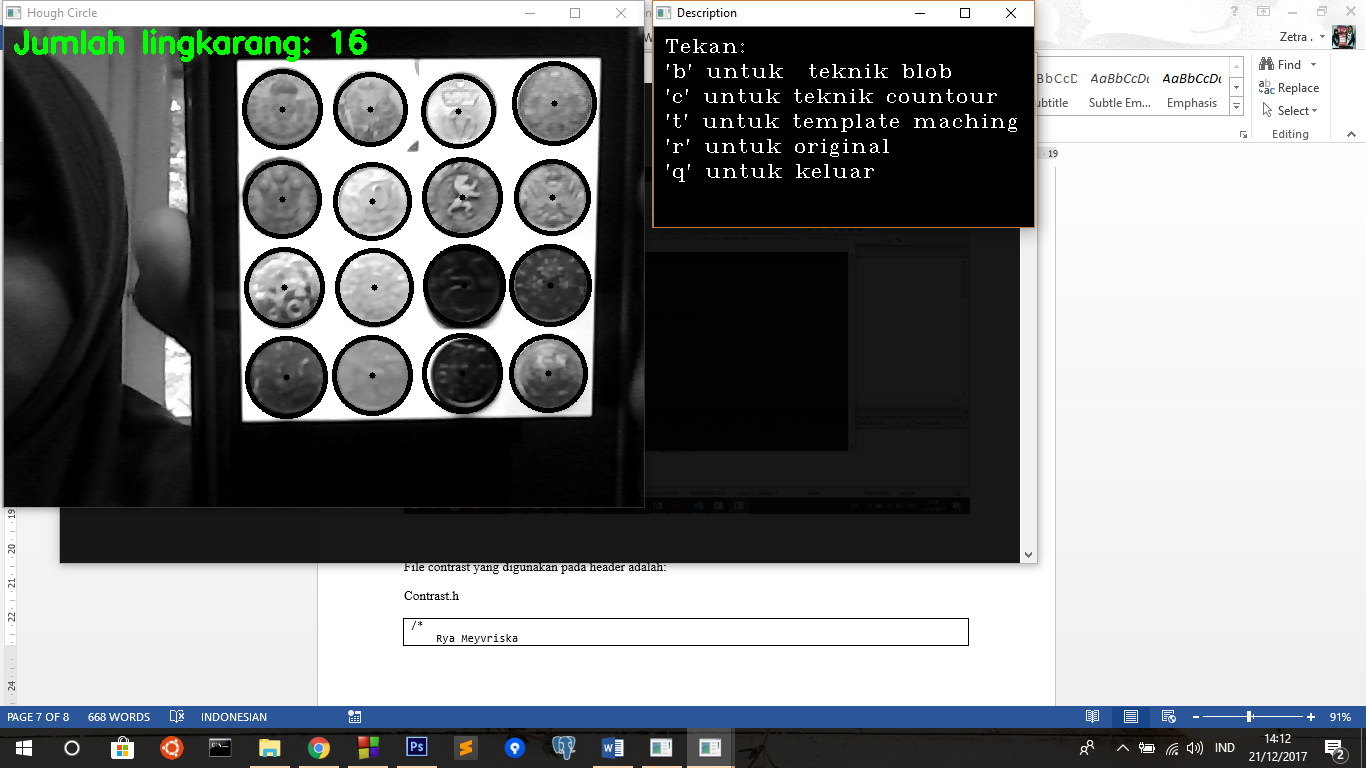
Header countur.cpp:

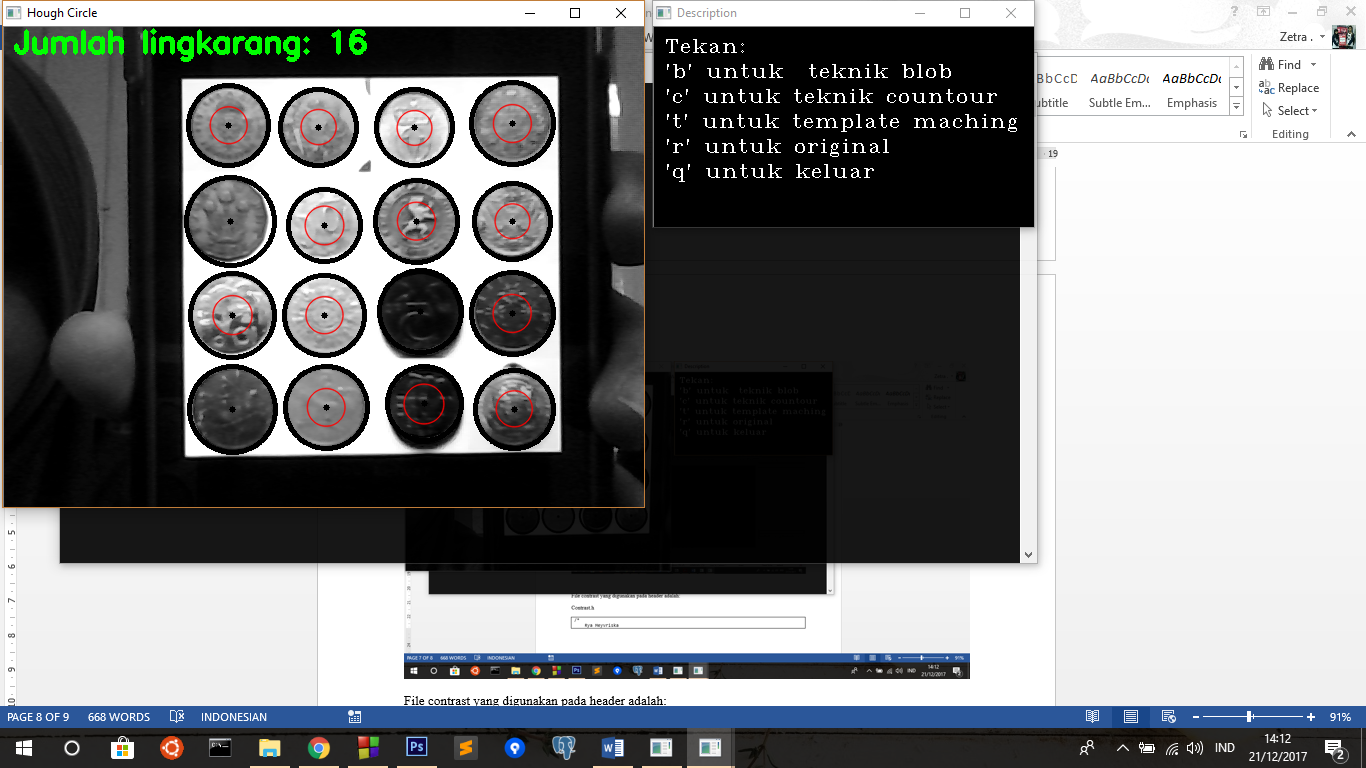
|  |
| --- |
| Mat countourTekhnik(Mat src\_gray){  int thresh = 100;  RNG rng(12345);  Mat canny\_output;  vector<vector<Point> > contours;  vector<Vec4i> hierarchy;  /// Detect edges using canny  Canny( src\_gray, canny\_output, thresh, thresh\*2, 3 );  /// Find contours  findContours( canny\_output, contours, hierarchy, CV\_RETR\_TREE, CV\_CHAIN\_APPROX\_SIMPLE, Point(0, 0) );  /// Draw contours  Mat drawing = Mat::zeros( canny\_output.size(), CV\_8UC3 );  for( int i = 0; i< contours.size(); i++ )  {  Scalar color = Scalar( rng.uniform(0, 255), rng.uniform(0,255), rng.uniform(0,255) );  drawContours( drawing, contours, i, color, 2, 8, hierarchy, 0, Point() );  }  return drawing;  } |

Header template\_matching.cpp

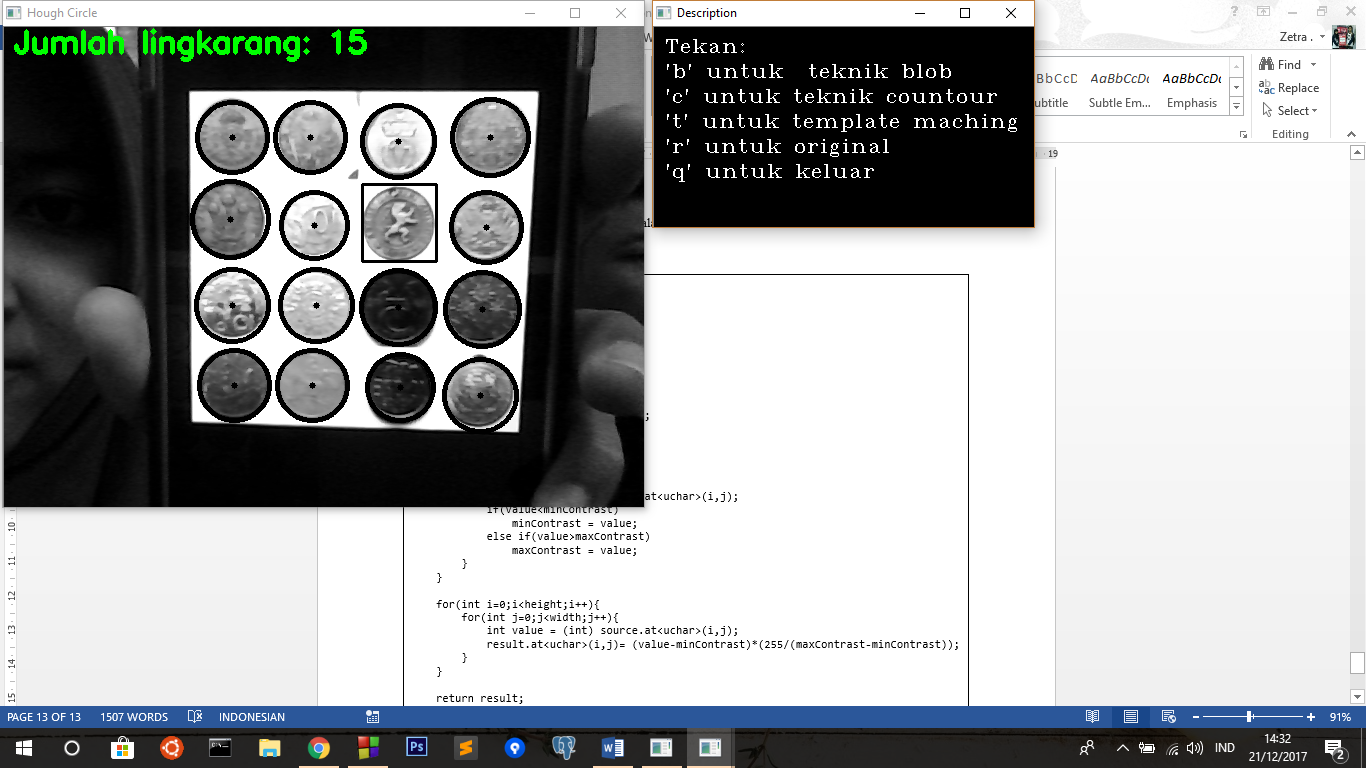
|  |
| --- |
| Mat templateMatchingTekhnik(Mat img, Mat templ){  /// If the type was not match  if(img.type()!=templ.type()){  cout<<"beda tipe"<<endl<<img.type()<<" "<<templ.type()<<endl;  return img;  }  /// Source image to display  Mat img\_display,result;  img.copyTo( img\_display );  int match\_method = 1;  /// Create the result matrix  int result\_cols = img.cols - templ.cols + 1;  int result\_rows = img.rows - templ.rows + 1;  result.create( result\_rows, result\_cols, CV\_32FC1 );  /// Do the Matching and Normalize  matchTemplate( img, templ, result, match\_method );  normalize( result, result, 0, 1, NORM\_MINMAX, -1, Mat() );  /// Localizing the best match with minMaxLoc  double minVal; double maxVal; Point minLoc; Point maxLoc;  Point matchLoc;  minMaxLoc( result, &minVal, &maxVal, &minLoc, &maxLoc, Mat() );  /// For SQDIFF and SQDIFF\_NORMED, the best matches are lower values. For all the other methods, the higher the better  if( match\_method == CV\_TM\_SQDIFF || match\_method == CV\_TM\_SQDIFF\_NORMED )  { matchLoc = minLoc; }  else  { matchLoc = maxLoc; }  /// Show me what you got  rectangle( img\_display, matchLoc, Point( matchLoc.x + templ.cols , matchLoc.y + templ.rows ), Scalar::all(0), 2, 8, 0 );  rectangle( result, matchLoc, Point( matchLoc.x + templ.cols , matchLoc.y + templ.rows ), Scalar::all(0), 2, 8, 0 );  return img\_display;  } |

Screenshoot :







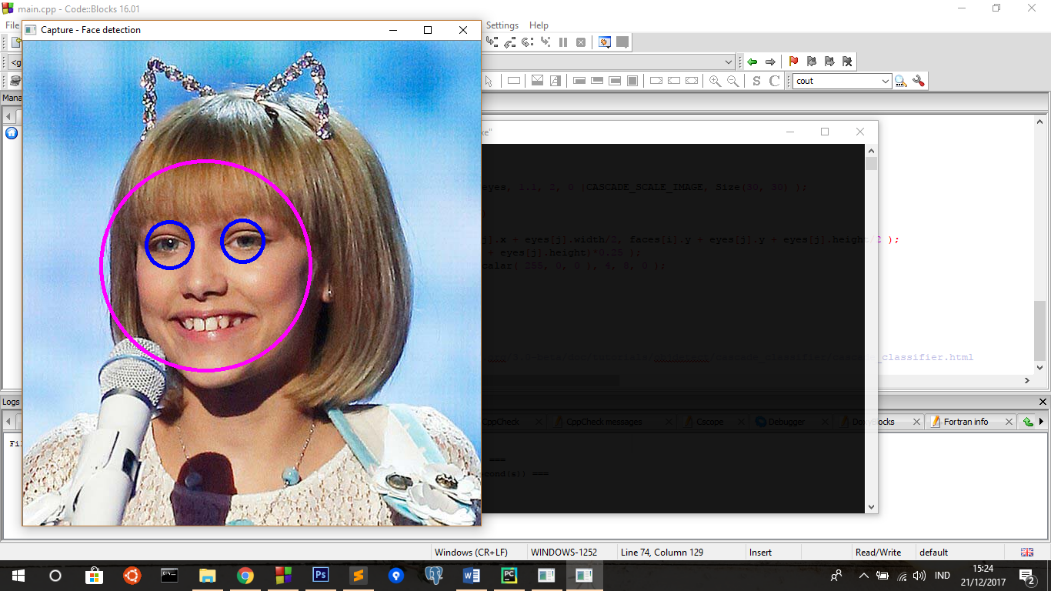


Dengan image untuk template method :

Task 1 – Poin 4

Buat program baru dengan memanfaatkan teknik Haar Classifier

|  |
| --- |
| #include "opencv2/objdetect/objdetect.hpp"  #include "opencv2/highgui/highgui.hpp"  #include "opencv2/imgproc/imgproc.hpp"  #include <iostream>  #include <stdio.h>  using namespace std;  using namespace cv;  /\*\* Function Headers \*/  void detectAndDisplay( Mat frame );  /\*\* Global variables \*/  String face\_cascade\_name = "haarcascade\_frontalface\_default.xml";  String eyes\_cascade\_name = "haarcascade\_eye.xml";  CascadeClassifier face\_cascade;  CascadeClassifier eyes\_cascade;  String window\_name = "Capture - Face detection";  int main()  {  Mat frame = imread("face.jpg");  //-- 1. Load the cascades  if( !face\_cascade.load( face\_cascade\_name ) ){ printf("--(!)Error loading face cascade\n"); return -1; };  if( !eyes\_cascade.load( eyes\_cascade\_name ) ){ printf("--(!)Error loading eyes cascade\n"); return -1; };  //-- 2. Read the video stream  if( frame.empty()){  printf(" --(!) No captured frame -- Break!");  }  //-- 3. Apply the classifier to the frame  detectAndDisplay( frame );  waitKey();  return 0;  }  void detectAndDisplay( Mat frame )  {  std::vector<Rect> faces;  Mat frame\_gray;  cvtColor( frame, frame\_gray, COLOR\_BGR2GRAY );  equalizeHist( frame\_gray, frame\_gray );  //-- Detect faces  face\_cascade.detectMultiScale( frame\_gray, faces, 1.1, 2, 0|CASCADE\_SCALE\_IMAGE, Size(30, 30) );  for( size\_t i = 0; i < faces.size(); i++ )  {  Point center( faces[i].x + faces[i].width/2, faces[i].y + faces[i].height/2 );  ellipse( frame, center, Size( faces[i].width/2, faces[i].height/2), 0, 0, 360, Scalar( 255, 0, 255 ), 4, 8, 0 );  Mat faceROI = frame\_gray( faces[i] );  std::vector<Rect> eyes;  //-- In each face, detect eyes  eyes\_cascade.detectMultiScale( faceROI, eyes, 1.1, 2, 0 |CASCADE\_SCALE\_IMAGE, Size(30, 30) );  for( size\_t j = 0; j < eyes.size(); j++ )  {  Point eye\_center( faces[i].x + eyes[j].x + eyes[j].width/2, faces[i].y + eyes[j].y + eyes[j].height/2 );  int radius = cvRound( (eyes[j].width + eyes[j].height)\*0.25 );  circle( frame, eye\_center, radius, Scalar( 255, 0, 0 ), 4, 8, 0 );  }  }  //-- Show what you got  imshow( window\_name, frame );  } |



Screenshoot:

Task 2 (opsional):

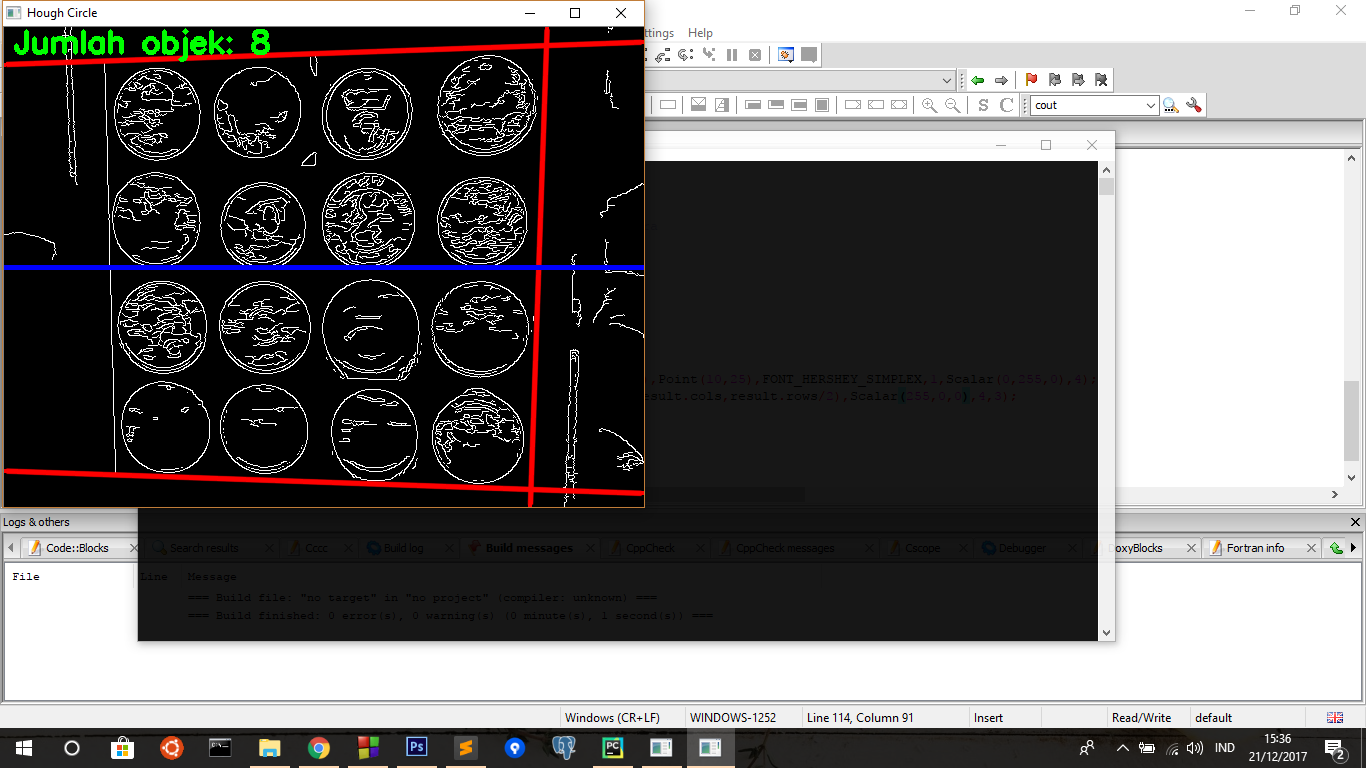
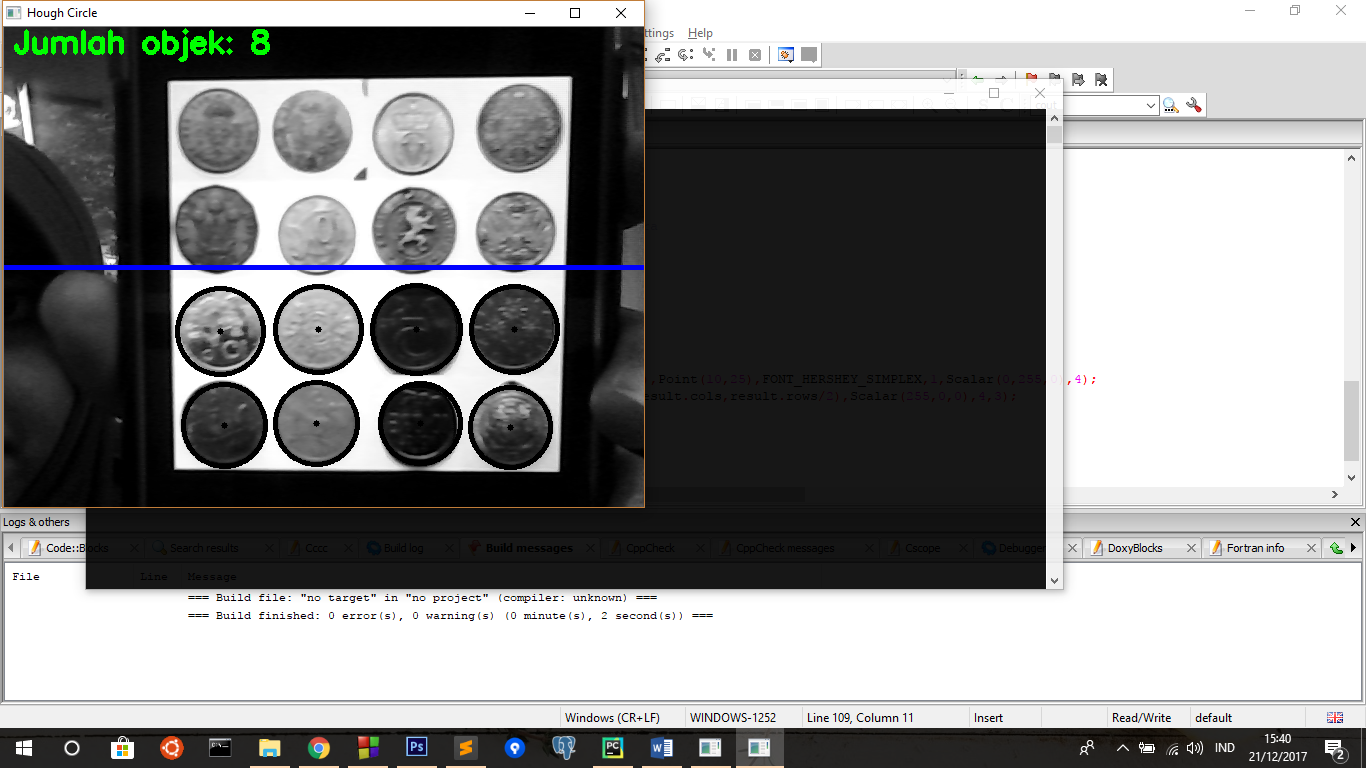
Buat sebuah program deteksi dengan input video(cam) dengan memanfaatkan fitur Hough Transform line untuk menghitung jumlah objek yang melewati garis yang diletakkan di tengah frame. (misalnya koin yang dilempar dari atas ke bawah) lalu munculkan jumlah hitung realtimenya pada layar.

Main.cpp :

|  |
| --- |
| /\*  Rya Meyvriska  G64164008  Task 2  \*/  #include "opencv2/highgui/highgui.hpp"  #include "opencv2/imgproc/imgproc.hpp"  #include <iomanip>  #include <iostream>  #include <sstream>  using namespace cv;  using namespace std;  class Circle{  private:  Mat dst,src;  int sum;  public:  void processHoughCircle(Mat picture){  src = picture.clone();  dst = Mat::zeros((src.rows/2)+1, src.cols, src.type());  sum = 0;  /// Ubah to Gray  if(src.channels()==3){  cvtColor(src,src,CV\_BGR2GRAY);  }  cvtColor(dst,dst,CV\_BGR2GRAY);  for(int i=0;i<dst.rows;i++){  for(int j=0;j<dst.cols;j++){  dst.at<uchar>(i,j) = src.at<uchar>((src.rows/2-1)+i,j);  }  }  /// img enhanchment untuk mereduce noise  GaussianBlur(dst,dst, Size(3, 3), 2, 2 );  medianBlur(dst,dst,3);  vector<Vec3f> circles;  /// LIBRARY HOUGH UNTUK MENDETEKSI LINGKARAN  HoughCircles( dst, circles, CV\_HOUGH\_GRADIENT, 1, 30, 200, 50, 0, 0 );  /// ALGORITMA MENGGAMBAR LINGKARAN  for( size\_t i = 0; i < circles.size(); i++ )  {  Point center(cvRound(circles[i][0]), cvRound(circles[i][1])+(src.rows/2));  int radius = cvRound(circles[i][2]);  circle( src, center, 3, Scalar(0,255,0), -1, 8, 0 );// circle center  circle( src, center, radius, Scalar(0,0,255), 3, 8, 0 );// circle outline  }  sum = circles.size();  }  Mat getSrc(){  return src;  }  int getSum(){  return sum;  }  };  Mat ProcessHoughLines(Mat src){  Mat dst, cdst;  Canny(src, dst, 50, 200, 3); ///EDGE DETECTION  cvtColor(dst, cdst, CV\_GRAY2BGR);  //cdst = dst.clone();  vector<Vec2f> lines, lines2;///VARIABLE UNTUK MENAMPUNG GAMBAR GARIS  /// LIBRARY HOUGH UNTUK MENDETEKSI GARIS  HoughLines(dst, lines, 1, CV\_PI/180, 150, 0, 0 );  /// ALGORITMA MENGGAMBAR GARIS  for( size\_t i = 0; i < lines.size(); i++ )  {  float rho = lines[i][0], theta = lines[i][1];  Point pt1, pt2;  double a = cos(theta), b = sin(theta);  double x0 = a\*rho, y0 = b\*rho;  pt1.x = cvRound(x0 + 1000\*(-b));  pt1.y = cvRound(y0 + 1000\*(a));  pt2.x = cvRound(x0 - 1000\*(-b));  pt2.y = cvRound(y0 - 1000\*(a));  line( cdst, pt1, pt2, Scalar(0,0,255), 3, CV\_AA);  }  return cdst;  }  int main(){  VideoCapture cap(0); // open the default camera  if(!cap.isOpened()) // check if we succeeded  return -1;  Circle c;  while(true)  {  Mat frame,result;  cap >> frame; // get a new frame from camera  flip(frame, frame,1); //agar tidak mirror  c.processHoughCircle(frame);  result = c.getSrc().clone();  cvtColor(result,result,CV\_GRAY2BGR);  result = ProcessHoughLines(frame);  ostringstream sum;  sum << c.getSum();  putText(result,"Jumlah objek: " + sum.str(),Point(10,25),FONT\_HERSHEY\_SIMPLEX,1,Scalar(0,255,0),4);  line(result,Point(0,result.rows/2),Point(result.cols,result.rows/2),Scalar(255,0,0),4,3);  imshow("Hough Circle", result);  if(waitKey(1) >= 0) break;  }  return 0;  } |

Screenshoot:

Dimana warna biru adalah garis tengah dari frame, lingkaran atau koin yang di bawah akan di hitung sedangkan yang diatas garis tidak akan dihitung.

  
  
Gambar Hanya lingkaran (baris 109 di hide), Line (baris 109 tidak di hide)

File contrast yang digunakan pada header program-program sebelumnya adalah:

Contrast.h

|  |
| --- |
| /\*  Rya Meyvriska  G64164008  Ilmu Komputer IPB  \*/  Mat contrast(Mat source){  Mat result;  result = source.clone();  int width = source.size().width;  int height = source.size().height;  int minContrast = 255;  int maxContrast = 0;  for(int i=0;i<height;i++){  for(int j=0;j<width;j++){  int value = (int) source.at<uchar>(i,j);  if(value<minContrast)  minContrast = value;  else if(value>maxContrast)  maxContrast = value;  }  }  for(int i=0;i<height;i++){  for(int j=0;j<width;j++){  int value = (int) source.at<uchar>(i,j);  result.at<uchar>(i,j)= (value-minContrast)\*(255/(maxContrast-minContrast));  }  }  return result;  } |