# İstanbul Medipol Üniversitesi
# Görüntü İşleme

Onur Karal H5170067

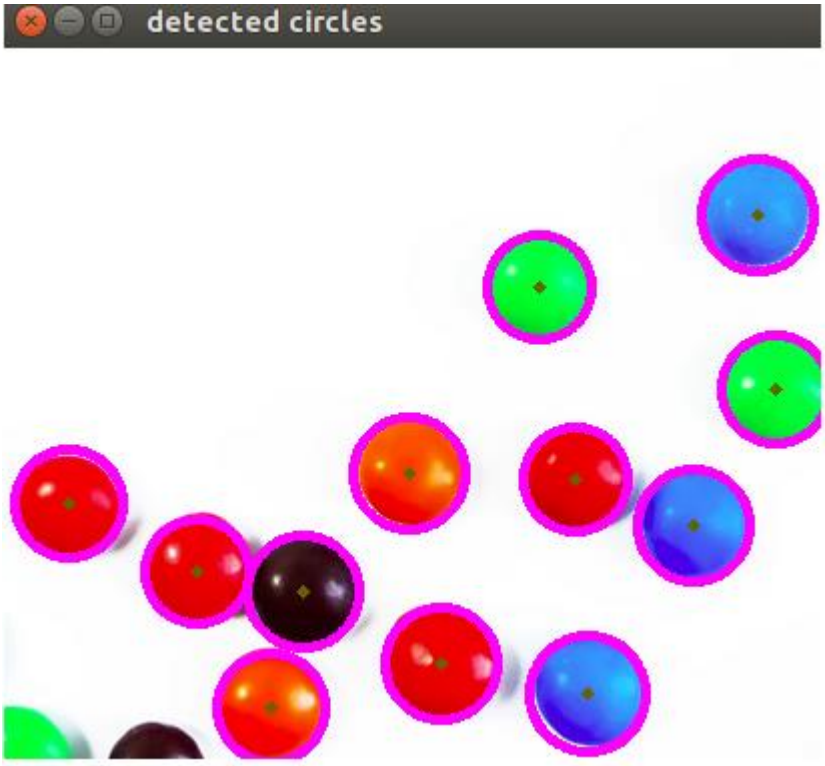
## Hough Line Transform ve Hough Circle Transform

### Hough Circle Transform;

Hafta 10 projesinin görüntü işlenen kısmı yani tuş'a tıklandığında yapılacak olan kısıma çemberleri algılayan ve etrafını ve merkezini belli eden kod eklendi.

```
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <iostream>
#include <stdio.h>

using namespace cv;

/** @function main */
int main(int argc, char** argv)
{
  Mat src, src_gray;

  /// Read the image
  src = imread( argv[1], 1 );

  if( !src.data )
    { return -1; }

  /// Convert it to gray
  cvtColor( src, src_gray, CV_BGR2GRAY );

  /// Reduce the noise so we avoid false circle detection
  GaussianBlur( src_gray, src_gray, Size(9, 9), 2, 2 );

  vector<Vec3f> circles;

  /// Apply the Hough Transform to find the circles
  HoughCircles( src_gray, circles, CV_HOUGH_GRADIENT, 1, src_gray.rows/8, 200, 100, 0, 0 );

  /// Draw the circles detected
  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 center
      circle( src, center, 3, Scalar(0,255,0), -1, 8, 0 );
      // circle outline
      circle( src, center, radius, Scalar(0,0,255), 3, 8, 0 );
   }

  /// Show your results
  namedWindow( "Hough Circle Transform Demo", CV_WINDOW_AUTOSIZE );
  imshow( "Hough Circle Transform Demo", src );

  waitKey(0);
  return 0;
}
```

```
    Mat src1, src_gray;
    src1 = conv.QImage2Mat( im1 );
    cvtColor(src1, src_gray, COLOR_BGR2GRAY);                                    use
      GaussianBlur( src_gray, src_gray, Size(9, 9), 2, 2 );      use of undeclared identifier 'src_gray
       std::vector<Vec3f> circles;                                               use of
       int maxtrehhold = ui->verticalSlider->value();
       HoughCircles( src_gray, circles, CV_HOUGH_GRADIENT, 1, src_gray.rows/maxtrehhold, 200, 100, 0, 0 );

       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 center
             circle( src1, center, 3, Scalar(0,255,0), -1, 8, 0 );
             // circle outline
             circle( src1, center, radius, Scalar(0,0,255), 3, 8, 0 );
          }
   im2 = conv.Mat2QImage(src1);
    pix2 = QPixmap::fromImage(im2);
    sahne2.addPixmap( pix2 );
    ui->graphicsView_2->setScene( &sahne2 );
}
```

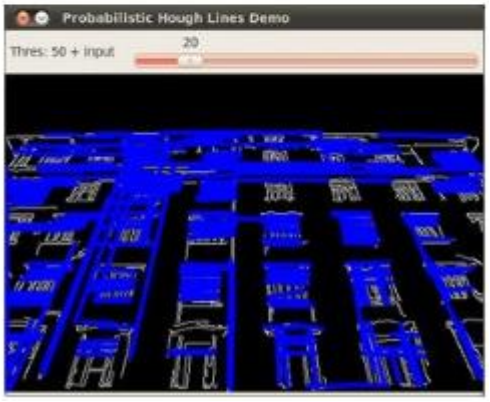



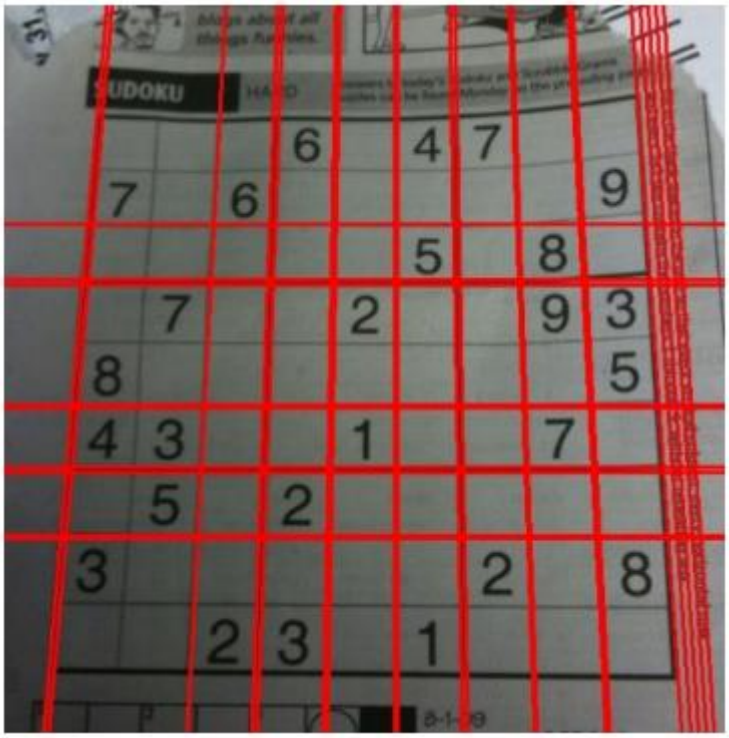
## Hough Line Transform;

Line yani dikey ve yatay çizgilerin algılarınıp çizilmesini sağlayan kod ise bu kısımıdan alınıp projeye eklendi.,

```
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"

#include <iostream>

using namespace cv;
using namespace std;

void help()
{
 cout << "\nThis program demonstrates line finding with the Hough transform.\n"
         "Usage:\n"
         "./houghlines <image_name>, Default is pic1.jpg\n" << endl;
}

int main(int argc, char** argv)
{
 const char* filename = argc >= 2 ? argv[1] : "pic1.jpg";

 Mat src = imread(filename, 0);
 if(src.empty())
 {
     help();
     cout << "can not open " << filename << endl;
     return -1;
 }

 Mat dst, cdst;
 Canny(src, dst, 50, 200, 3);
 cvtColor(dst, cdst, CV_GRAY2BGR);

 #if 0
  vector<Vec2f> lines;
  HoughLines(dst, lines, 1, CV_PI/180, 100, 0, 0 );

  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);
  }
 #else
  vector<Vec4i> lines;
  HoughLinesP(dst, lines, 1, CV_PI/180, 50, 50, 10 );
  for( size_t i = 0; i < lines.size(); i++ )
  {
    Vec4i l = lines[i];
    line( cdst, Point(l[0], l[1]), Point(l[2], l[3]), Scalar(0,0,255), 3, CV_AA);
  }
 #endif
 imshow("source", src);
 imshow("detected lines", cdst);

 waitKey();

 return 0;
}
```

```
{
    Mat src1;
    Mat  dst1,cdst;
    src1 = conv.QImage2Mat( im1 );
    Canny(src1, dst1, 50, 200, 3);
     cvtColor(dst1, cdst, CV_GRAY2BGR);
    int lineslider = ui->verticalSlider->value();
  std::vector<Vec4i> lines;
  HoughLinesP(dst1, lines, 1, CV_PI/lineslider, 100, 0, 0 );
  for( size_t i = 0; i < lines.size(); i++ )
  {
    Vec4i l = lines[i];
    line( cdst, Point(l[0], l[1]), Point(l[2], l[3]), Scalar(0,0,255), 3, CV_AA);
  }

      im2 = conv.Mat2QImage(cdst);
       pix2 = QPixmap::fromImage(im2);
       sahne2.addPixmap( pix2 );

       ui->graphicsView_2->setScene( &sahne2 );

}
```






Sonrasında iki görüntü işleme işlemi için slider eklendi ve bunun sayesinde küçük ve büyük şekillerin algılanması sağlandı.

```
void MainWindow::on_verticalSlider_valueChanged(int value)
{

}
```

```
int lineslider = ui->verticalSlider->value();
```

```
int maxtrehhold = ui->verticalSlider->value();
```