**PROBLEM STATEMENT:**

Implement Canny-edge detector algorithm from scratch using C++ to detect the edges in an Image. Neat Documentation is expected with Code, Explanation, Input, and Output Image

**SOLUTION:**

The Canny Edge detector  was developed by John F. Canny in 1986. Also known to many as the optimal detector, the Canny algorithm aims to satisfy three main criteria:

* **Low error rate:** Meaning a good detection of only existent edges.
* **Good localization:** The distance between edge pixels detected and real edge pixels have to be minimized.
* **Minimal response:** Only one detector response per edge.

**CODE:**

#include "[opencv2/imgproc.hpp](https://docs.opencv.org/3.4/dd/d46/imgproc_8hpp.html)"

#include "[opencv2/highgui.hpp](https://docs.opencv.org/3.4/d4/dd5/highgui_8hpp.html)"

#include <iostream>

using namespace [cv](https://docs.opencv.org/3.4/d2/d75/namespacecv.html);

[Mat](https://docs.opencv.org/3.4/d3/d63/classcv_1_1Mat.html)src, src\_gray;

[Mat](https://docs.opencv.org/3.4/d3/d63/classcv_1_1Mat.html)dst, detected\_edges;

int lowThreshold = 0;

const int max\_lowThreshold = 100;

const int ratio = 3;

const int kernel\_size = 3;

const char\* window\_name = "Edge Map";

static void CannyThreshold(int, void\*)

{

[blur](https://docs.opencv.org/3.4/d4/d86/group__imgproc__filter.html#ga8c45db9afe636703801b0b2e440fce37)( src\_gray, detected\_edges, [Size](https://docs.opencv.org/3.4/dc/d84/group__core__basic.html#ga346f563897249351a34549137c8532a0)(3,3) );

[Canny](https://docs.opencv.org/3.4/dd/d1a/group__imgproc__feature.html#ga04723e007ed888ddf11d9ba04e2232de)( detected\_edges, detected\_edges, lowThreshold, lowThreshold\*ratio, kernel\_size );

dst = [Scalar::all](https://docs.opencv.org/3.4/d1/da0/classcv_1_1Scalar__.html#ac1509a4b8454fe7fe29db069e13a2e6f)(0);

src.[copyTo](https://docs.opencv.org/3.4/d3/d63/classcv_1_1Mat.html#a33fd5d125b4c302b0c9aa86980791a77)( dst, detected\_edges);

[imshow](https://docs.opencv.org/3.4/d7/dfc/group__highgui.html#ga453d42fe4cb60e5723281a89973ee563)( window\_name, dst );

}

int main( intargc, char\*\* argv )

{

[CommandLineParser](https://docs.opencv.org/3.4/d0/d2e/classcv_1_1CommandLineParser.html)parser( argc, argv, "{@input | fruits.jpg | input image}" );

src = [imread](https://docs.opencv.org/3.4/d4/da8/group__imgcodecs.html#ga288b8b3da0892bd651fce07b3bbd3a56)( [samples::findFile](https://docs.opencv.org/3.4/d6/dba/group__core__utils__samples.html#ga3a33b00033b46c698ff6340d95569c13)( parser.get<[String](https://docs.opencv.org/3.4/d1/d8f/classcv_1_1String.html)>( "@input" ) ), [IMREAD\_COLOR](https://docs.opencv.org/3.4/d4/da8/group__imgcodecs.html#gga61d9b0126a3e57d9277ac48327799c80af660544735200cbe942eea09232eb822) ); // Load an image

if( src.[empty](https://docs.opencv.org/3.4/d3/d63/classcv_1_1Mat.html#abbec3525a852e77998aba034813fded4)() )

{

std::cout<< "Could not open or find the image!\n" << std::endl;

std::cout<< "Usage: " <<argv[0] << " <Input image>" << std::endl;

return -1;

}

dst.[create](https://docs.opencv.org/3.4/d3/d63/classcv_1_1Mat.html#a55ced2c8d844d683ea9a725c60037ad0)( src.[size](https://docs.opencv.org/3.4/d3/d63/classcv_1_1Mat.html#a146f8e8dda07d1365a575ab83d9828d1)(), src.[type](https://docs.opencv.org/3.4/d3/d63/classcv_1_1Mat.html#af2d2652e552d7de635988f18a84b53e5)() );

[cvtColor](https://docs.opencv.org/3.4/d8/d01/group__imgproc__color__conversions.html#ga397ae87e1288a81d2363b61574eb8cab)( src, src\_gray, [COLOR\_BGR2GRAY](https://docs.opencv.org/3.4/d8/d01/group__imgproc__color__conversions.html#gga4e0972be5de079fed4e3a10e24ef5ef0a353a4b8db9040165db4dacb5bcefb6ea) );

[namedWindow](https://docs.opencv.org/3.4/d7/dfc/group__highgui.html#ga5afdf8410934fd099df85c75b2e0888b)( window\_name, [WINDOW\_AUTOSIZE](https://docs.opencv.org/3.4/d7/dfc/group__highgui.html#ggabf7d2c5625bc59ac130287f925557ac3acf621ace7a54954cbac01df27e47228f) );

[createTrackbar](https://docs.opencv.org/3.4/d7/dfc/group__highgui.html#gaf78d2155d30b728fc413803745b67a9b)( "Min Threshold:", window\_name, &lowThreshold, max\_lowThreshold, CannyThreshold );

CannyThreshold(0, 0);

[waitKey](https://docs.opencv.org/3.4/d7/dfc/group__highgui.html#ga5628525ad33f52eab17feebcfba38bd7)(0);

return 0;

}

**EXPLANATION:**

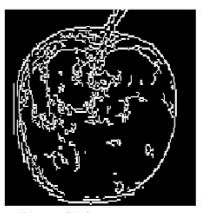
1. Create some needed variables
2. Loads the source image
3. Create a matrix of the same type and size of src (to be dst)
4. Convert the image to grayscale (using the function [cv::cvtColor](https://docs.opencv.org/3.4/d8/d01/group__imgproc__color__conversions.html#ga397ae87e1288a81d2363b61574eb8cab) )
5. Create a window to display the results
6. Create a Trackbar for the user to enter the lower threshold for Canny detector
7. Check the CannyThreshold function
8. fill a dst image with zeros (meaning the image is completely black).
9. Finally, use the function to map only the areas of the image that are identified as edges (on a black background).To copy the src image onto dst. However, it will only copy the pixels in the locations where they have non-zero values. Since the output of the Canny detector is the edge contours on a black background, the resulting dst will be black in all the area but the detected edges.
10. Display the result

**OUTPUT:**

**Input image:**

****

**Output image:**

****