HÁSKÓLINN Í REYKJAVÍK

COMPUTER VISION

T-869-COMP

Assignment 2

Sindri Þór Harðarson

sindrih18@ru.is

Instructor

Torfi Þórhallsson

November 28, 2023

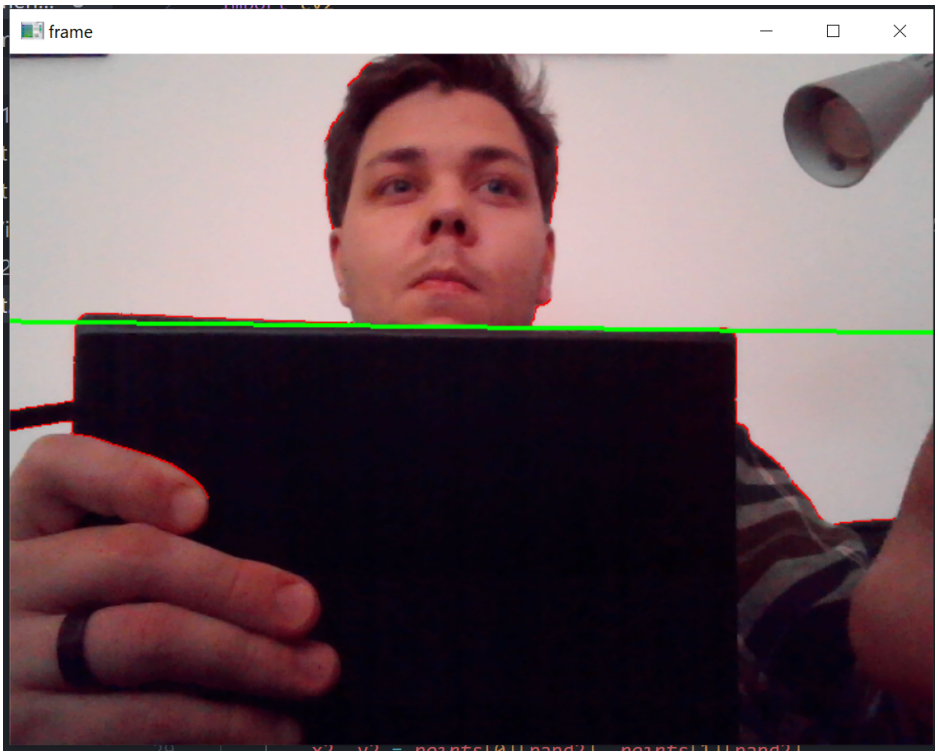

Figure 1: Showing edge detection

https://github.com/SindriTh/ComputerVision

1 Questions

1.1 How well your straight line detector follows the edge of a sheet of paper moved across the camera's field of view.

It does so pretty well, but also gets quite easily "distracted". This is likely something that could be fine tuned either with the CANNY function or the RANSAC parameters.
My parameters were chosen as they were a nice balance in different lighting scenarios. This could however be tuned further for specific environments.

1.2 How well does it detect other straight lines in your environment?

Sometimes too well. Sometimes it gets a little confused if there are a lot of straight lines in the scene, this is however difficult to remove with parameters, as the edges are quite similar.

1.3 The processing time for one video frame or image.

My implementation takes around 300-350ms to process with 200 iterations and an inlier threshold of 600. This threshold is quite high, but lowering it introduced quite a bit of noise. This is likely due to the fact that the CANNY function is providing too many edges.

2 Code

```python
# Sindri Þór Harðarson
import numpy as np
import cv2
import time

FRAME_WIDTH = 640
FRAME_HEIGHT = 480
CANNY_THRESHOLD1 = 400
CANNY_THRESHOLD2 = 600
RANSAC_ITERATIONS = 200
RANSAC_THRESHOLD = 10 # The distance from the line that a point can be and still be consi
RANSAC_INLIER_THRESHOLD = 600 # The number of inliers required to stop the RANSAC algorit

def getRANSAC(points, iterations, threshold, inlierThreshold):
    bestLine = None
    bestInliers = 0
    if len(points[0]) < 2:
        return None

    print(f"Points: {len(points[0])}")
    for i in range(iterations):
        # Get 2 random points
        rand1 = np.random.randint(0, len(points[0]))
        rand2 = np.random.randint(0, len(points[0]))
        # Make sure they are different
        while rand1 == rand2:
            rand2 = np.random.randint(0, len(points[0]))
        # Get the coordinates of the points
        x1, y1 = points[0][rand1], points[1][rand1]
        x2, y2 = points[0][rand2], points[1][rand2]
        # Calculate the line between the points
        # y = mx + b
        if x1 == x2:
            # Handle vertical line
            # For a vertical line, slope is infinity, and 'b' is the x-coordinate
            m = np.inf
            b = x1
        else:
            # Calculate the line between the points
            # y = mx + b
            m = (y2 - y1) / (x2 - x1)
            b = y1 - (m * x1)
        # Calculate distance from line for each point
        inliers = 0
        for j in range(len(points[0])):
            # Get coordinates of point
```

```
            x = points[0][j]
            y = points[1][j]
            # Calculate distance from line
            distance = abs((m * x) - y + b) / np.sqrt((m**2) + 1)
            # Check if inlier
            if distance < threshold:
                inliers += 1
        # Check if best fit
        if inliers > bestInliers:
            bestInliers = inliers
            bestLine = (m, b)
            # Check if enough inliers
            if inliers > inlierThreshold:
                print(f"Found line with {inliers} inliers")
                break
    return bestLine

def main():
    cap = cv2.VideoCapture(0)
    cap.set(cv2.CAP_PROP_FRAME_WIDTH, FRAME_WIDTH)
    cap.set(cv2.CAP_PROP_FRAME_HEIGHT, FRAME_HEIGHT)
    cap.set(cv2.CAP_PROP_AUTO_WB, 0.0) # Disable automatic white balance
    cap.set(cv2.CAP_PROP_WB_TEMPERATURE, 4200) # Set manual white balance temperature to


    while True:
        # Starting time
        start_frame_time = time.time()
        ret, frame = cap.read()

        # Canny edge detection
        edges = cv2.Canny(frame, CANNY_THRESHOLD1, CANNY_THRESHOLD2)

        # Process edges and RANSAC
        edgeArray = np.where(edges == 255)
        frame[edgeArray] = [0, 0, 255]
        ransac = getRANSAC(edgeArray, RANSAC_ITERATIONS, RANSAC_THRESHOLD, RANSAC_INLIER_

        if ransac is not None:
            # Draw line
            m, b = ransac
            y1 = 0
            x1 = int(m * y1 + b)
            y2 = FRAME_WIDTH
            x2 = int(m * y2 + b)
            cv2.line(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)

        print(f"Processing time: {time.time() - start_frame_time:.4f}")
```

```
        cv2.imshow('frame', frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()

if __name__ == "__main__":
    main()
```