LAB 3: Tension Detection of Rolling Metal Sheet

Tension Detection of Rolling Metal Sheet

Date: 2024-May-09

Author: Ignacio Manteca Escudero 22320052

Github: Ignacio Github

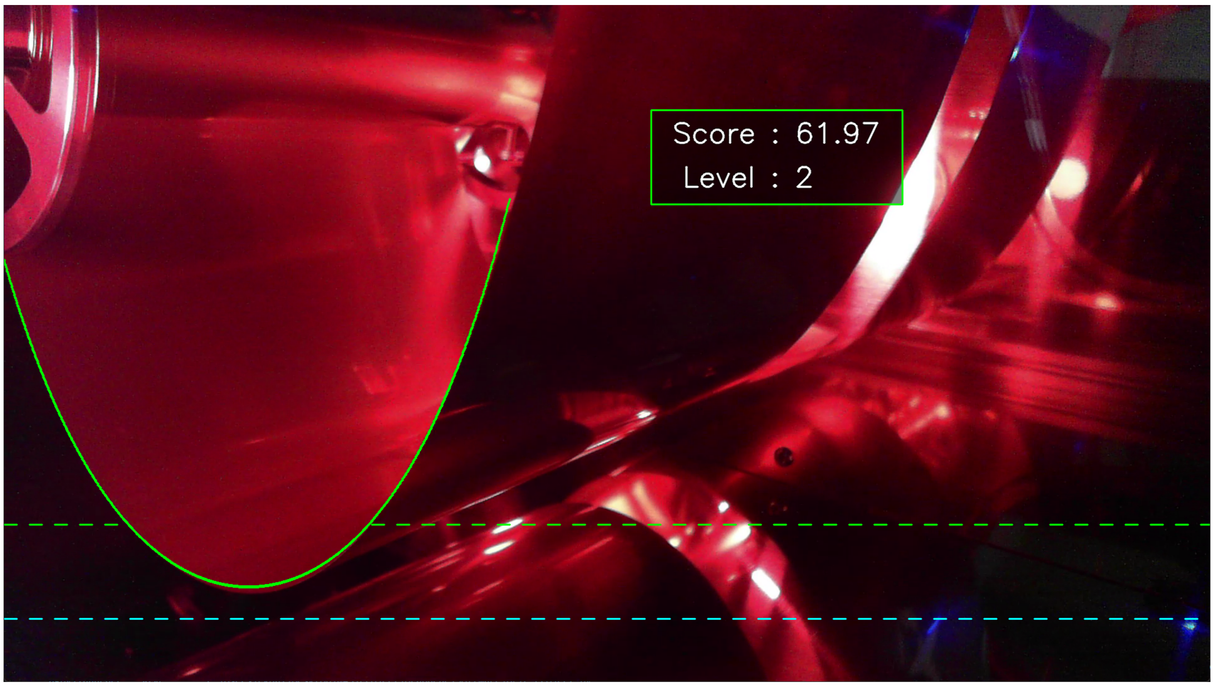
Introduction

1. Objective

Goal: This is a simplified industrial problem for designing a machine vision system that can detect the level of tension in the rolling metal sheet.

The tension in the rolling process can be derived by measuring the curvature level of the metal sheet with the camera.

The surface of the chamber and the metal sheet are both specular reflective that can create virtual objects in the captured images. You need to design a series of machine vision algorithms to clearly detect the edge of the metal sheet and derive the curvature and tension level.



2. Preparation

For this lab we have used a computer with visual studio installed, as well as python and the nummpy and openCV libraries. As for the sources used, a video is attached, from which different screenshots have been taken to test the programme.

Software Installation

- OpenCV 4.9, Visual Studio 2022
- Python 3.11.1

Dataset

Two different data sets have been used. First, the images provided in the simple image data set were used. Once we managed to complete the exercise with satisfactory results for this dataset, we moved on to the advanced image data set, which improved the programme in certain aspects. Finally, a separate program was also developed, which works directly with the video and not only with the images obtained from it.

Dataset link:

Download the test images of

- Simple dataset

- Challenging dataset
- Video

Algorithm

1. Overview

The first thing we need to know is that we have two different programs. One is used to open the images from the dataset ("Lab3.py")and the other uses the same analysis method but is designed to analyse video ("Lab3_video.py"). The algorithm used and the processing method are the same, only the source is different.
Therefore, I will only explain one of them as they are the same.

2. Procedure

Open image and separate channels

In the first step of our code we read the image we want to use from the dataset. Because the original images are very large and we want to rescale them, we use a series of ratios to keep the proportions of the original image. Because our program works with image distances, if we rescale without keeping the proportions of the original image, this can change our result.

After that, because the images in the dataset have a red illumination, I decided to analyse the different outputs obtained from the RGB channels. And I saw that the red channel would be easier to analyse, as it removes the noise and highlights some areas of the metal plate which makes it easier to detect the edges and thus the tension level.

Filtering

We then apply a medium filter, which removes noise and blurs everything except the edges of the metal plates, to make them easier to detect and prevent Canny from detecting false edges.

Select ROI

This is because the image is very wide and there are various elements that can make it difficult to detect the edge of the sheet metal. We choose an ROI that allows us to select only the specific area where we need to analyse the image to obtain the information of interest.

Edge Detection

Once we have selected our ROI to work with, we use the canny() function to detect the edges and create a mask the size of the original image, to which we then add the result of drawing our contours after using the findContours() function.
We must remember that when we draw the results in our mask, we must add the difference in height. This is because the borders have been calculated on the region of interest and the mask has the size of the original image.

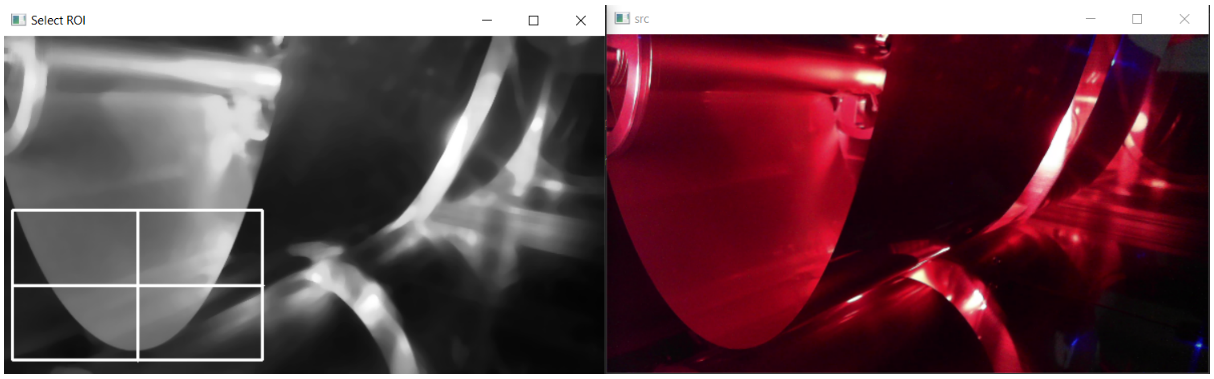


Figure 1. On the left we can see the result of the image after selecting the red channel and applying the median filter, we can also see how the white area is selected to be our ROI. On the right we have the original image after resizing.

Point detection

Now that we have detected the edge of the metal plate, we want to know the height of the lowest point of the plate so that we know how much tension it has. To do this, we use a for loop that goes through all the points of the edge and compares them to find the lowest point. And after comparing it with the levels suggested by the exercise, we determine the level of tension that the metal plate is at.

Remember that because of the resize we did in the beginning on the original image. Now we must rescale all these calculated points so that they are proportional to the original ones.

Further down in the code you can see a couple more processes which are mainly graphical functions to display the results in a convenient way.

Result and Discussion

1. Final Result

The results obtained with both datasets are satisfactory and meet the requirements. They clearly show the score, which is the number of pixels between the point and the end of the image. They also show the stress level and the lines that define each level. The programme works correctly for all the images provided in both the simple and complex datasets.

As for the video. There are moments in the video where the edge detection fails a bit and detects edges that are not of the metal plate. But these can be easily corrected by adjusting the ROI to a narrower area centred on the metal plate and this way correct results are obtained throughout the video.

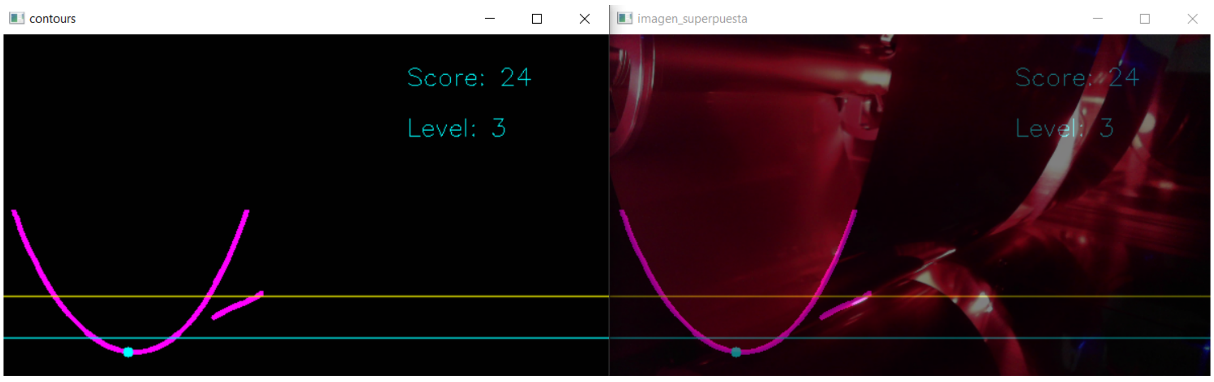


Figure 2. On the right is the original image, overlaid with the edges calculated by the Canny function. And on the left we have a look at what our edge detection is getting.

2. Discussion

The results obtained work correctly with 100% accuracy when using images as a data set. And for video, if the ROI is correctly selected, the program correctly detects the voltage level practically throughout the video.

Conclusion

In this case, it is a good programme to detect the edges and the stress level of the metal plates. To create a better program, the edge detection could be improved and limited to one, so that only the edges of the metal plate are detected, eliminating all the external edges that are not to be calculated.
A system to complete the curve formed by the metal plate could also be used. Due to the lighting, gaps can sometimes appear in the detected curve.

Appendix

```
#Lab3.py Detecting using images from Dataset
import numpy as np
import cv2 as cv

# Load image
img = cv.imread('C:/Users/Ignacio/source/repos/DLIP/PyOpenCvE
original_heigth, original_with = img.shape[:2]
new_heigth = int((600 / original_with) * original_heigth)
img2 = cv.resize(img, (600,new_heigth))

cv.namedWindow('src', cv.WINDOW_AUTOSIZE)
cv.imshow('src', img2)

#Separate original images into colours chanels RGB
canal_azul, canal_verde, canal_rojo = cv.split(img2)

#Apply median filter to reduce noise and maintain edges
median = cv.medianBlur(canal_rojo,11)

#Select ROI
roi = cv.selectROI('Select ROI', median, fromCenter=False, sh
x, y, w, h = roi
roi_selected = median[y:y+h, x:x+w]


cv.destroyAllWindows()

#Detect the edges of the curve
```

```
edges = cv.Canny(roi_selected, 120, 240,None, 3)
contours, _=cv.findContours(edges, cv.RETR_EXTERNAL, cv.CHAIN_
#Create a mask to draw contours
filled_image = np.ones_like(img2)
cv.drawContours(filled_image[y:y+h, x:x+w], contours, -1, (25

#Paint horizontal lines indicating the levels of tension
max_y = None
max_point_y = None

lvl1 = original_heigth - 250 #Level 1: >250px from the bottom
lvl1 = int((new_heigth / original_heigth) * lvl1)
lvl2 = original_heigth - 120 #Level 2: 120~250 px from the bo
lvl2 = int((new_heigth / original_heigth) * lvl2)

cv.line(filled_image, (0, lvl1), (600, lvl1), (0, 255, 255),
cv.line(filled_image, (0, lvl2), (600, lvl2), (255, 255, 0),

# FInd the lowest point from the detected edge
for contorno in contours:
    for point in contorno.squeeze():
        if max_y is None or point[1] > max_y:
            max_y = point[1]
            max_point_y = point

# Determine the level of tension
max_y = max_y + y
if(max_y<=lvl1):
    cv.circle(filled_image[y:y+h, x:x+w], max_point_y, 5, (0,
    color = (0, 255, 0)
    lvl = 1
elif(max_y>lvl1 and max_y<lvl2):
    cv.circle(filled_image[y:y+h, x:x+w], max_point_y, 5, (0,
    color=(0, 255, 255)
    lvl = 2
else:
    cv.circle(filled_image[y:y+h, x:x+w], max_point_y, 5, (25
    color=(255, 255, 0)
    lvl = 3

# Desing grafic elements and print
#The score value is calculated respect the new image size, th
max_y = new_heigth - max_y
text1 = f"Score: {max_y}"
text2 = f"Level: {lvl}"
cv.putText(filled_image, text1, (400,50), cv.FONT_HERSHEY_SIM
cv.putText(filled_image, text2, (400,100), cv.FONT_HERSHEY_SI

cv.namedWindow('contours', cv.WINDOW_AUTOSIZE)
cv.imshow('contours', filled_image)

alpha = 0.5
imagen_superpuesta = cv.addWeighted(img2, 1 - alpha, filled_i
cv.namedWindow('imagen_superpuesta', cv.WINDOW_AUTOSIZE)
cv.imshow('imagen_superpuesta', imagen_superpuesta)

cv.waitKey(0)
cv.destroyAllWindows()
```

```
#Lab3_video.py Detecting using video as Dataset
import numpy as np
import cv2 as cv

# Load image
# img = cv.imread('C:/Users/Ignacio/source/repos/DLIP/PyOpenC
# original_heigth, original_with = img.shape[:2]
# new_heigth = int((600 / original_with) * original_heigth)
# img2 = cv.resize(img, (600,new_heigth))

# cv.namedWindow('src', cv.WINDOW_AUTOSIZE)
# cv.imshow('src', img2)

#Video
cap = cv.VideoCapture('C:/Users/Ignacio/source/repos/DLIP/PyO

if not cap.isOpened():
    print("Error al abrir el video")
    exit()

ret, img = cap.read()
original_heigth, original_with = img.shape[:2]
new_heigth = int((600 / original_with) * original_heigth)
img2 = cv.resize(img, (600,new_heigth))

#Select ROI
roi = cv.selectROI('Select ROI', img2, fromCenter=False, show
x, y, w, h = roi

cv.destroyAllWindows()

while cap.isOpened():
    ret, img = cap.read()

    img2 = cv.resize(img, (600,new_heigth))

    cv.imshow('Video', img2)

    #Separate original images into colours chanels RGB
    canal_azul, canal_verde, canal_rojo = cv.split(img2)

    #Apply median filter to reduce noise and maintain edges
    median = cv.medianBlur(canal_rojo,11)

    roi_selected = median[y:y+h, x:x+w]

    #Detect the edges of the curve
    edges = cv.Canny(roi_selected, 120, 240,None, 3)
    contours, _=cv.findContours(edges, cv.RETR_EXTERNAL, cv.C
    #Create a mask to draw contours
    filled_image = np.ones_like(img2)
    cv.drawContours(filled_image[y:y+h, x:x+w], contours, -1,

    #Paint horizontal lines indicating the levels of tension
    max_y = None
    max_point_y = None

    lvl1 = original_heigth - 250 #Level 1: >250px from the bo
    lvl1 = int((new_heigth / original_heigth) * lvl1)
    lvl2 = original_heigth - 120 #Level 2: 120~250 px from th
    lvl2 = int((new_heigth / original_heigth) * lvl2)

    cv.line(filled_image, (0, lvl1), (600, lvl1), (0, 255, 25
    cv.line(filled_image, (0, lvl2), (600, lvl2), (255, 255,

    # FInd the lowest point from the detected edge
    for contorno in contours:
        for point in contorno.squeeze():
            if max_y is None or point[1] > max_y:
                max_y = point[1]
                max_point_y = point

    # Determine the level of tension
    max_y = max_y + y
    if(max_y<=lvl1):
        cv.circle(filled_image[y:y+h, x:x+w], max_point_y, 5,
        color = (0, 255, 0)
        lvl = 1
    elif(max_y>lvl1 and max_y<lvl2):
        cv.circle(filled_image[y:y+h, x:x+w], max_point_y, 5,
        color=(0, 255, 255)
        lvl = 2
    else:
        cv.circle(filled_image[y:y+h, x:x+w], max_point_y, 5,
        color=(255, 255, 0)
        lvl = 3

    # Desing grafic elements and print
    #The score value is calculated respect the new image size
    max_y = original_heigth - max_y
    text1 = f"Score: {max_y}"
    text2 = f"Level: {lvl}"
    cv.putText(filled_image, text1, (400,50), cv.FONT_HERSHEY
    cv.putText(filled_image, text2, (400,100), cv.FONT_HERSHE

    cv.namedWindow('contours', cv.WINDOW_AUTOSIZE)
    cv.imshow('contours', filled_image)

    alpha = 0.5
    imagen_superpuesta = cv.addWeighted(img2, 1 - alpha, fill
    cv.namedWindow('imagen_superpuesta', cv.WINDOW_AUTOSIZE)
    cv.imshow('imagen_superpuesta', imagen_superpuesta)

    if cv.waitKey(70) & 0xFF == ord('q'):
        break
```

/** @about LAB3 Tension Detection of Rolling Metal Sheet