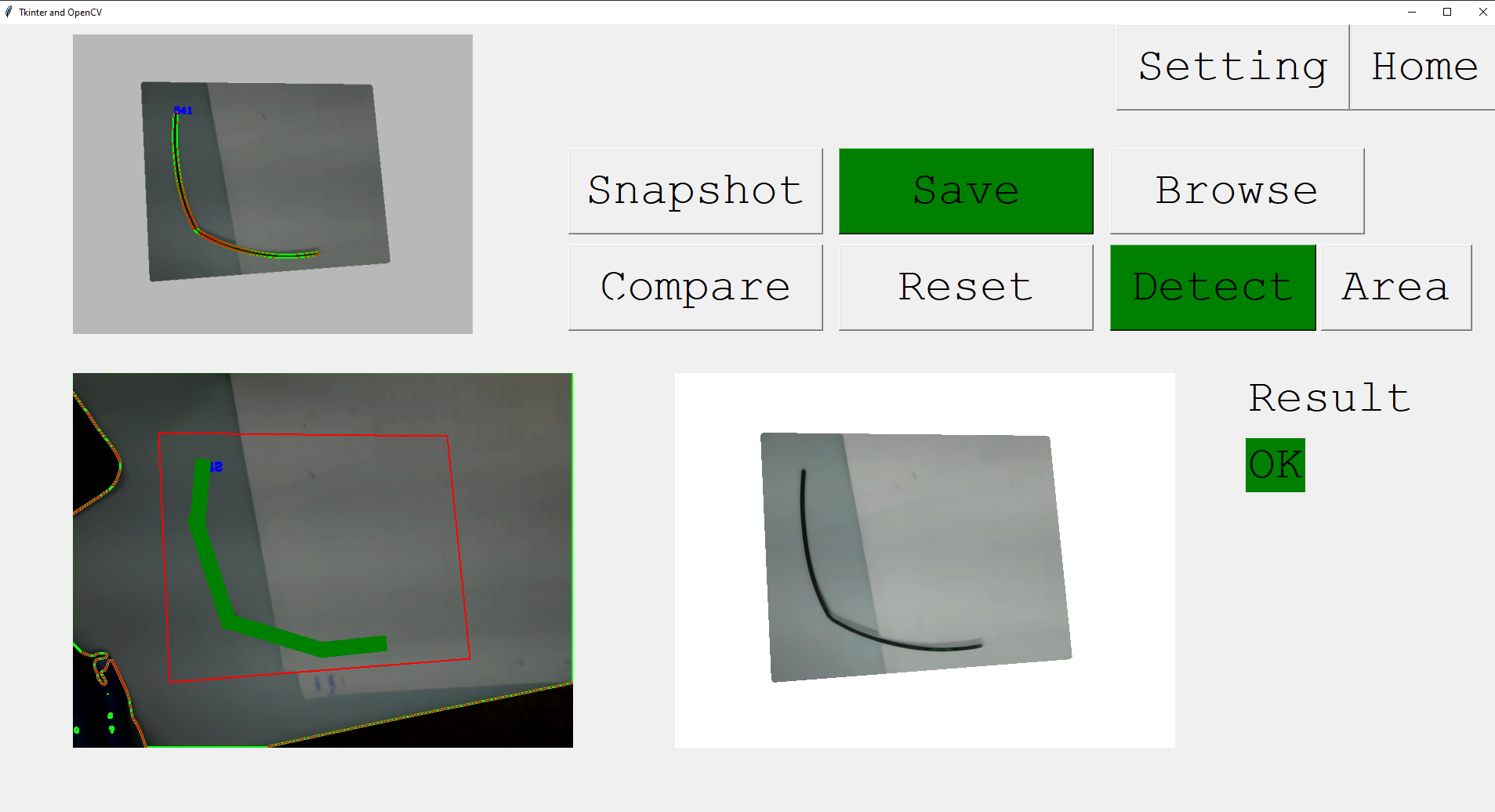
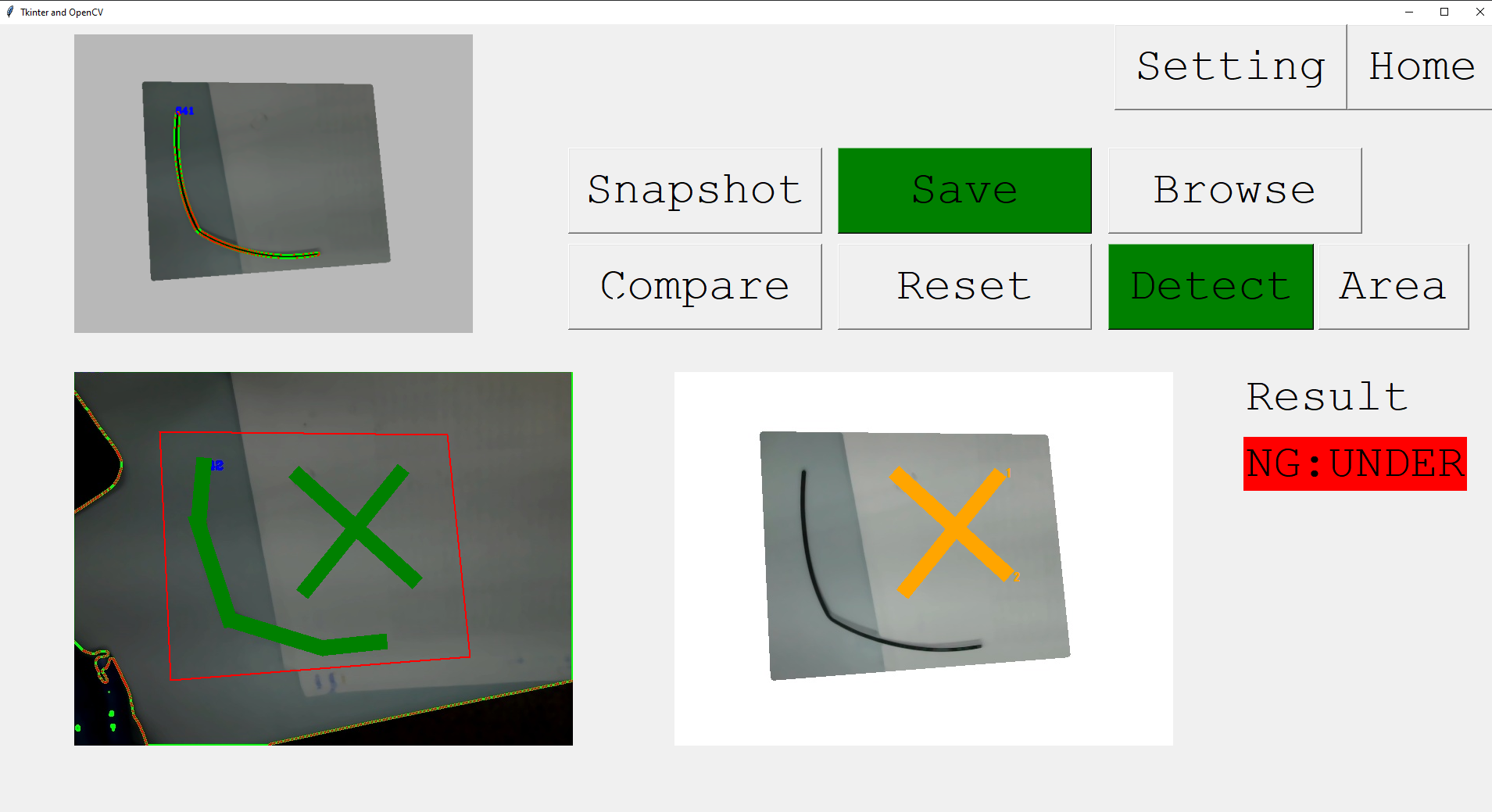
# Line Detection

## Introduction

**Line Detection** is the python software which use to check completed line in image. Currently, we use this software to detect sealing on car’s part. This software work on both of Local PC and Mamos that you can easily capture and control GPIO input/output.



**(No error Case)**



**(Error case: Because of none object in detect line [green])**

## Developer/ User Guide

### Requirements

Python 3.5.3

Pip 20.3.3

OS: Windows/ Linux/ Debian 9.5 (Mamos supported)

### Libraries

cycler==0.10.0  
imutils==0.5.3  
kiwisolver==1.1.0  
matplotlib==3.0.3  
numpy==1.18.5  
opencv-python==3.4.10.37  
Pillow==7.2.0  
pyparsing==2.4.7  
python-dateutil==2.8.1  
PyYAML==5.3.1  
scipy==1.4.1  
six==1.15.0  
shapely==1.7.1

(requirement.txt)

### Installation

#### # Windows

1. To download the code.

>> git clone xxx.git

>> cd <clone dir>

1. Check your python version.

>> python3 –version

Python 3.5.3

>> pip –version  
Pip 20.3.3

1. Create python environments.

python3 –m pip install –user virtualenv

python3 –m venv venv

1. Run virtual environment.

source venv/bin/activate

1. To install python libraries.

>> pip install –r requirements.txt

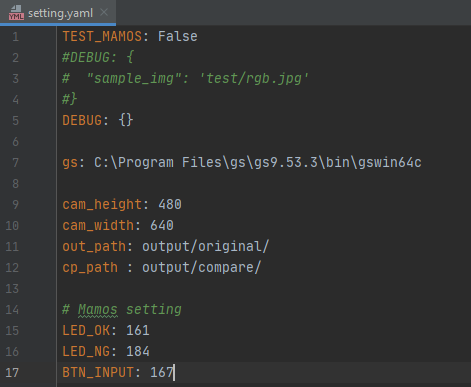
1. pip install –r requirements.txt
2. **ghost script install:** <https://www.ghostscript.com/download/gsdnld.html>

#### # Debian

\* Important lib  
> sudo apt-get update  
  
> sudo apt-get install python3-dev python3-pip python3-venv  
  
> (pip3 install setuptools)  
  
opencv  
  
> sudo apt install build-essential cmake git pkg-config libgtk-3-dev \  
 libavcodec-dev libavformat-dev libswscale-dev libv4l-dev \  
 libxvidcore-dev libx264-dev libjpeg-dev libpng-dev libtiff-dev \  
 gfortran openexr libatlas-base-dev python3-dev python3-numpy \  
 libtbb2 libtbb-dev libdc1394-22-dev  
  
pillow  
  
> sudo apt-get install libtiff-dev libjpeg-dev zlib1g-dev libfreetype6-dev liblcms2-dev libwebp-dev tcl-dev tk-dev python-tk  
  
shapely  
  
> sudo apt-get install libgeos-dev  
  
\* ASUS GPIO  
> git clone https://github.com/TinkerBoard/gpio\_lib\_python.git  
  
> cd ASUS\_GPIO\_PYTHON\_PATH/gpio/  
  
> sudo python3 setup.py install  
>  
\* Install python library  
  
> cd embed\_gui\_imagePATH  
  
\* Install opencv & opencv-contrib: https://linuxize.com/post/how-to-install-opencv-on-debian-10/  
> pip3 install scikit-build  
  
> pip3 install cython  
  
> pip3 install numpy==1.18.5  
  
> pip3 install opencv-python==3.4.10.37  
  
> pip3 install -r requirements.txt  
  
\* Install ghostscript:   
> sudo apt-get install ghostscript

### Configuration

Setting.yaml



1. TEST\_MAMOS: True for run with Mamos, otherwise to run on PC
2. DEBUG: You can use sample image instead of image from webcam
3. gs: This is importance when running on PC. You need to Install ghost script (<https://www.ghostscript.com/download/gsdnld.html>), then edit file path here
4. cam\_height/ cam\_width: Setting pixel size of camera source
5. out\_path/ cp\_path: These are directories path to “save snap image (original)” and “compare image“ \*\* you don’t need to change it
6. Mamos setting: These are GPIO setting

### Run Software

1. Run virtual environment.

source venv/bin/activate

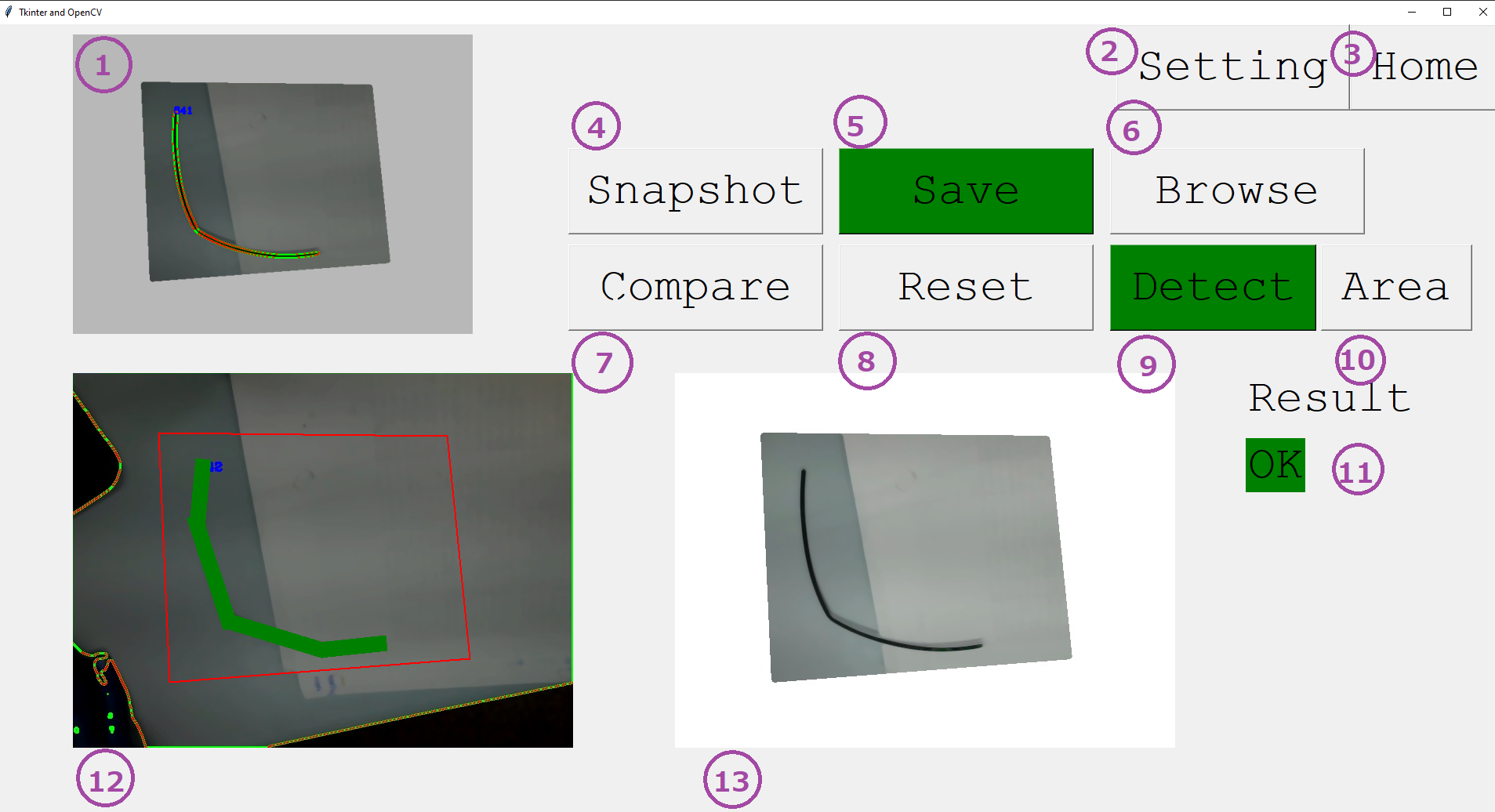
1. Run software

(venv) >> python3 Pages\_gui\_black\_detection.py

### Coding definition: [\_build/html/index.html](file:///C:/Users/thirat/Documents/git/TinkerBoard_line_detection35/_build/html/index.html)

### GUI

#### Home page



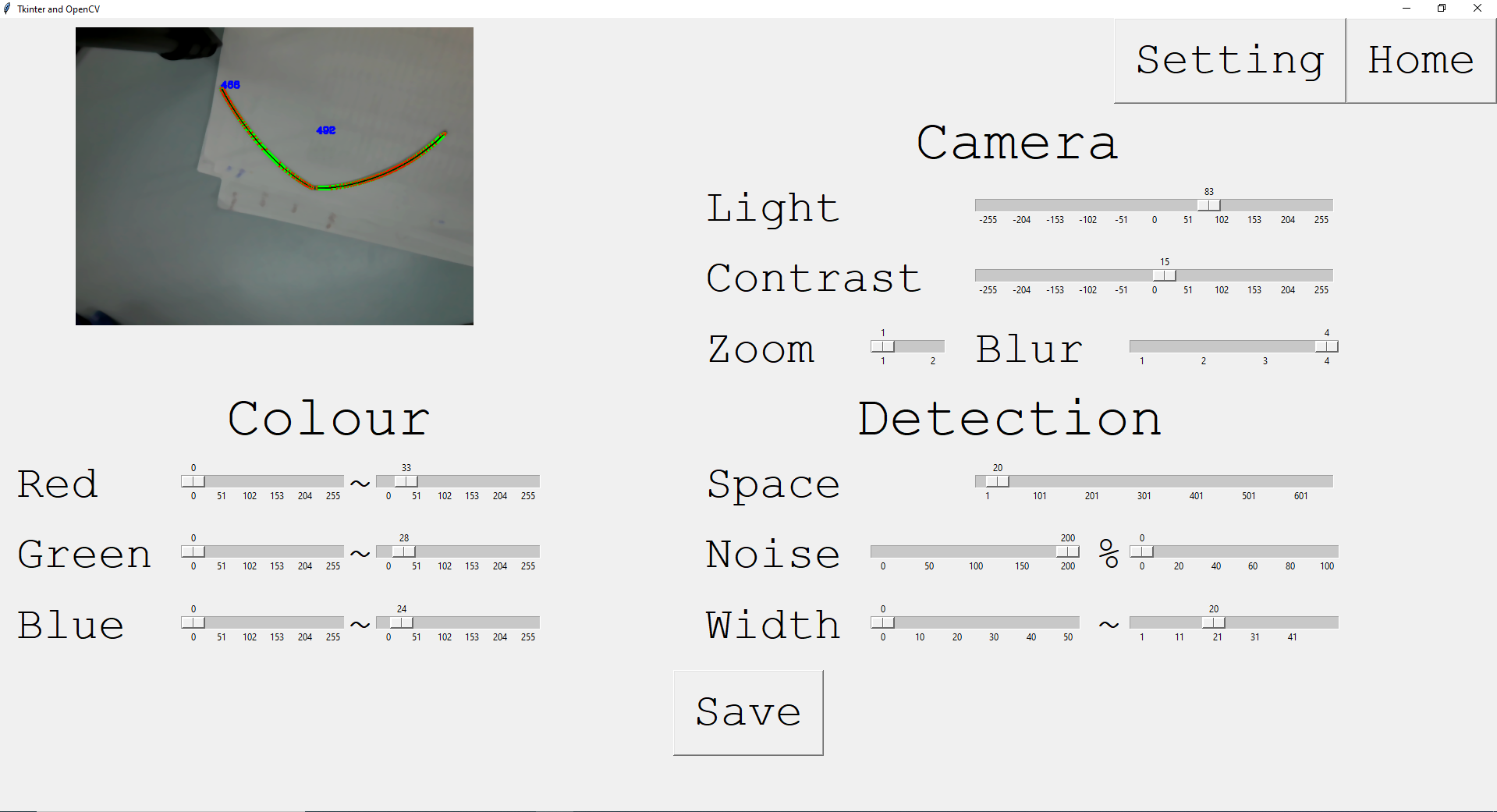
1. **Video source:** To show image from camera source
2. **Setting:** To go to **“Setting page”**
3. **Home:** To go to **“Home page”**
4. **Snapshot button:** To save original image to Snapshot image(12)
5. **Save button:** To save original image and drawing data to directories.

Drawing data: ./data/

Original image: ./output/original

1. **Browse button:** To load drawing data from json file, then show the image on the Snapshot image(12)
2. **Compare button:** To get comparing result, then show it in Output image(13)
3. **Reset button:** To clear all drawing and variable
4. **Detect button:** To change drawing mode to “Detect” mode (green line)
5. **Area button:** To change drawing mode to “Area” mode (red line)
6. **Result:** To show comparing result
7. **Snapshot image:** To show image from snapshot
8. **Output image:** To show image from comparing result

#### Setting page



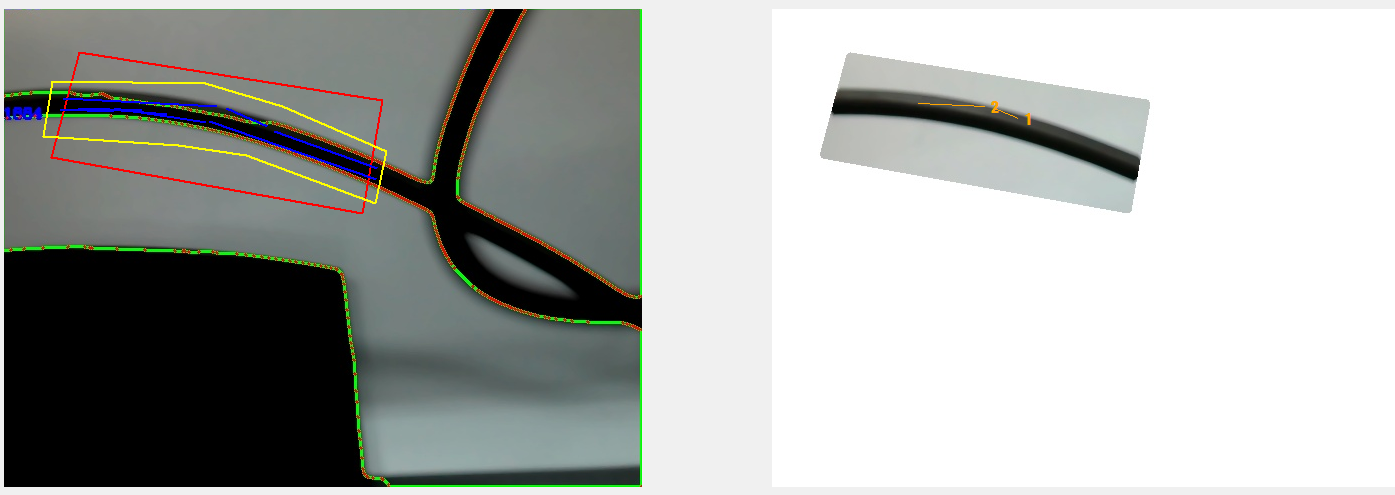
# Camera

1. **Light:** To config light in image
2. **Contrast:** To config contrast in image
3. **Zoom:** To zoom in/zoom out
4. **Blur:** To make image blur

# Colour: To define range of RGB detection by minimum – maximum values

# Detection

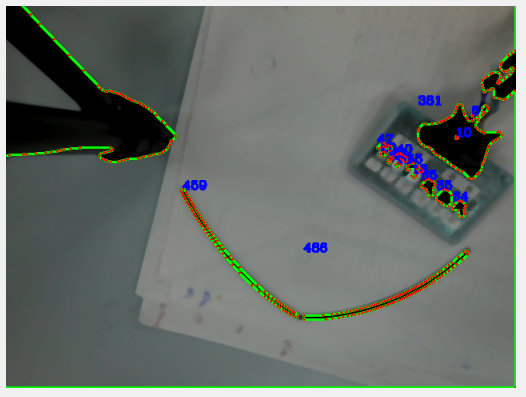
1. **Space (px):** To set threshold of space between “green line” and “object line”

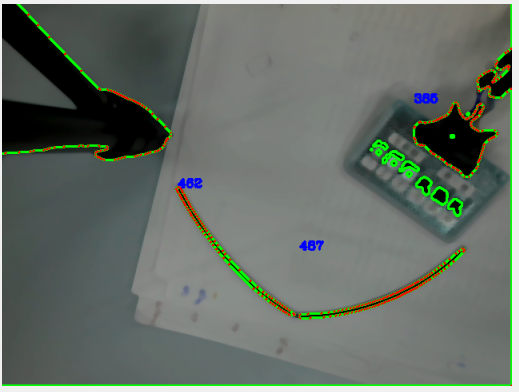


**(Space = 1)**

**(Space = 20)**

1. **Noise:** There have 2 value to set hear
2. Minimum length of object line detection

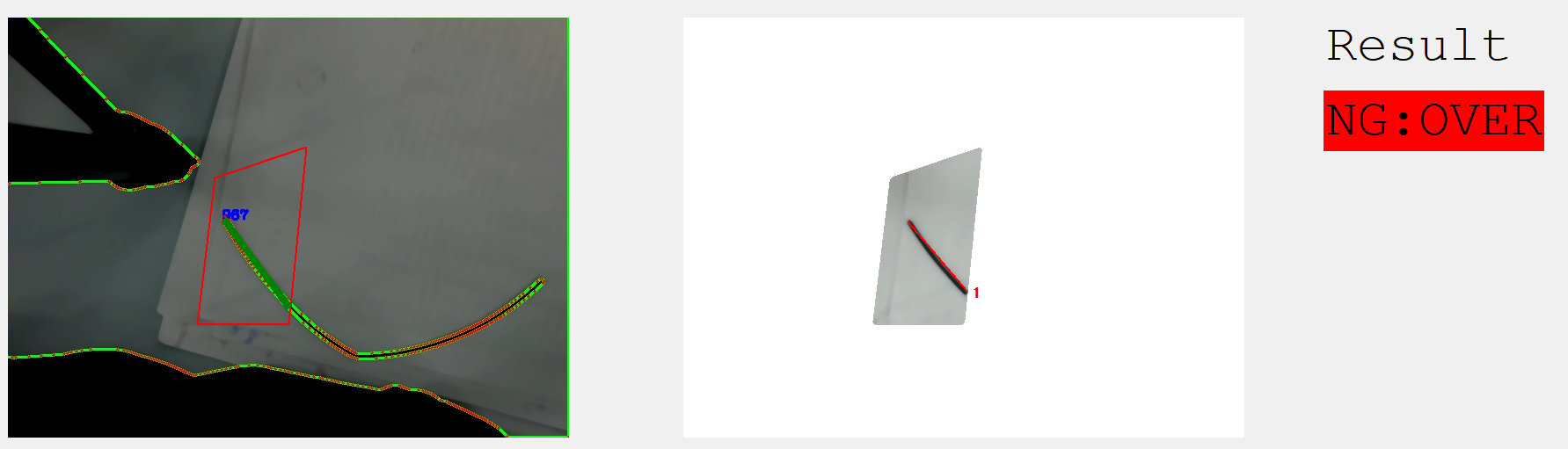


**(Minimum length = 10)**

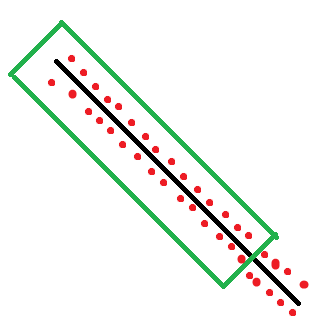
**(Minimum length = 100)**

**\*\* Note: you can config this parameter to remove some noise**

1. Error of object detection line (%)

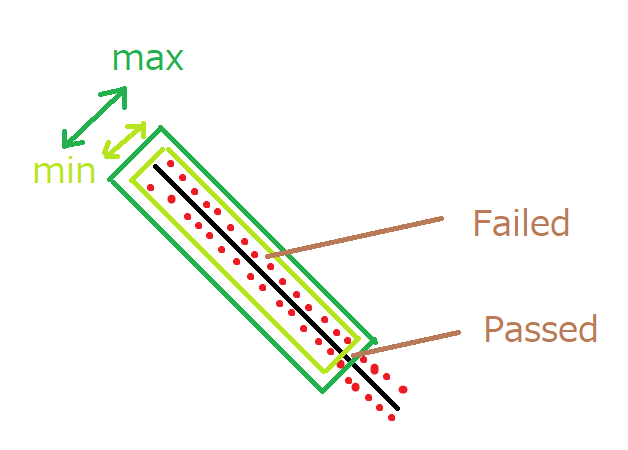


**(Error of object detection line = 5%)**

****

**For example: Object line respresent by red dot = 100 dot. If outside detect line have red dot more than (Error of object detection line = 5%), that object line will be error because of overflow**

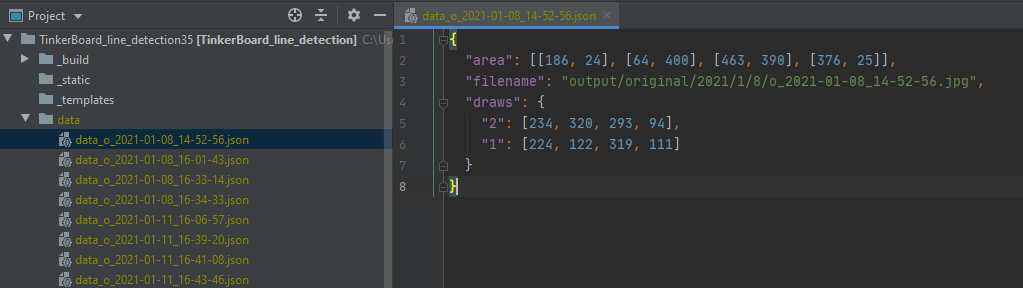
1. **Width (px):** This parameters are use to control min/max width of “detect line”



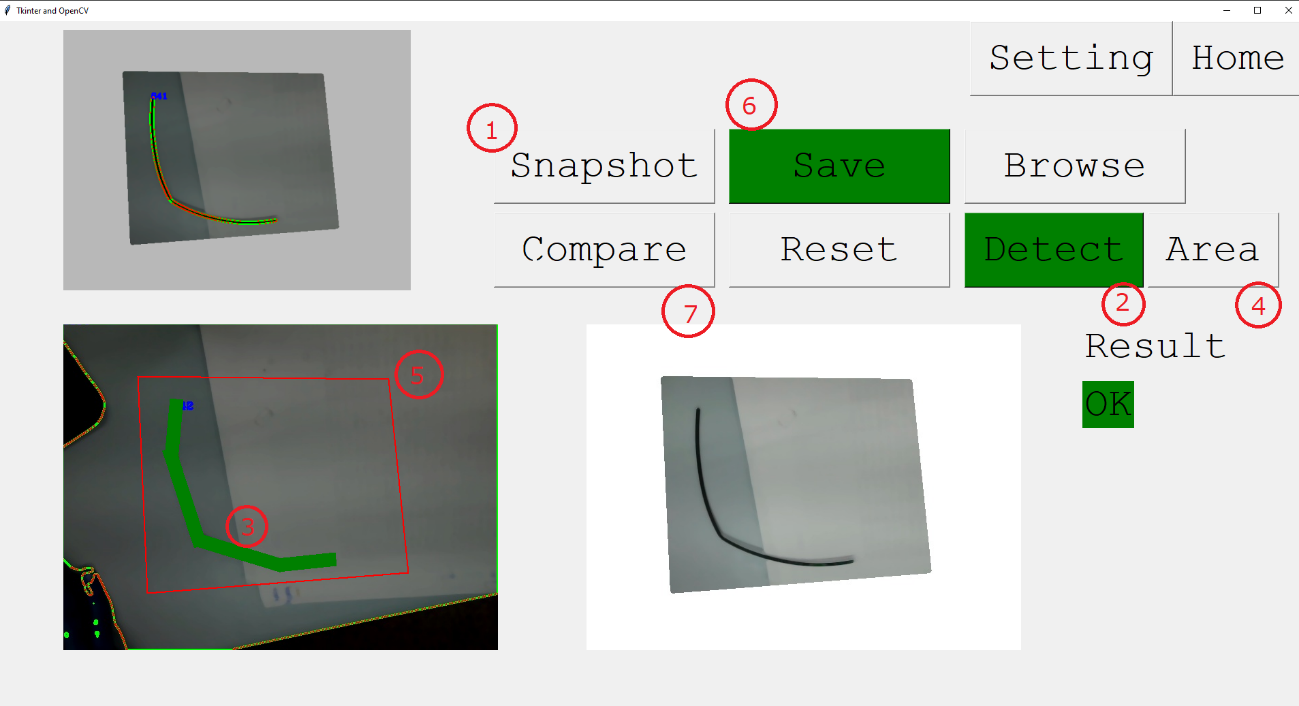
(This software will count red dot if **max width < dot < max width)**

### How to use

When you start this software, it will automatically call the last json data that contains of “image name” and “drawing data” if json file is exits.



**(Example of json data)**

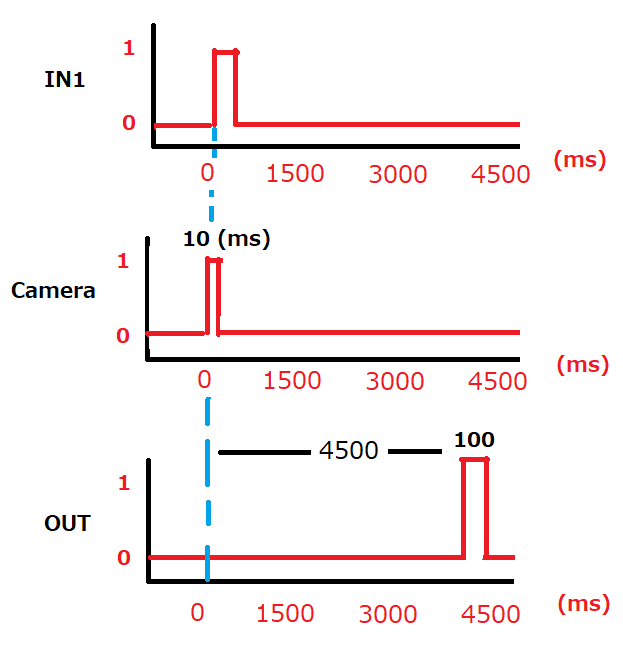


**(How to compare image)**

1. **Snapshot**
2. **Click Detect Button**
3. **Draw Detect line**
4. **Click Area Button**
5. **Draw Area line**
6. **Click Save Button**
7. **Click Compare Button/ Push Input pin (follow setting.yaml)**

### Running Time

After you press button (PIN 167), it will capture current image from camera, then it will use the image in crop area to calculate similarity and show the result to output pin in 4500 millisecond (crop area = 640x480 pixel). This calculate time is depend on length of object line & detect line in crop area



(Crop area = 640x480 pixel)

## Reference

1. Save canvas: Ghostscript  
   <https://www.ghostscript.com/>
2. Design GUI: Tkinter  
   <https://docs.python.org/3/library/tkinter.html>
3. Comparing geometry data  
   <https://shapely.readthedocs.io/en/stable/manual.html>