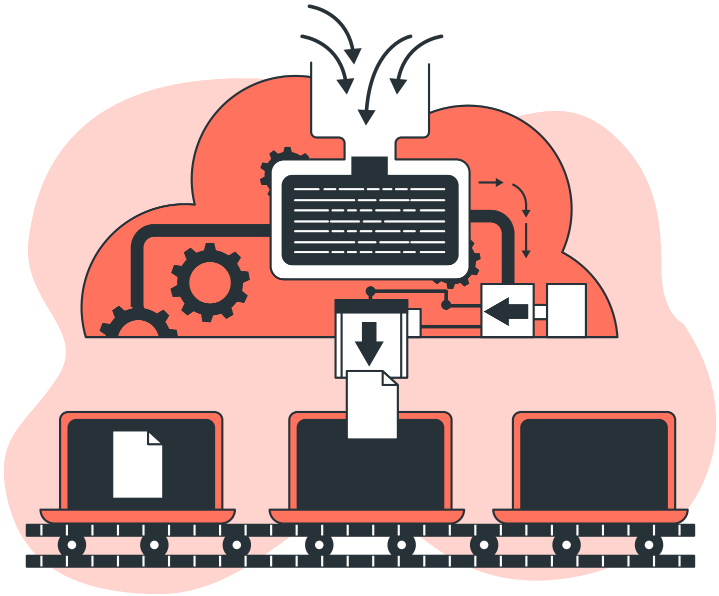
Automated Citrus Sorting System

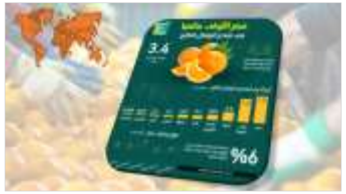
Dr. Mohamed Azzam

Automated Citrus Sorting System

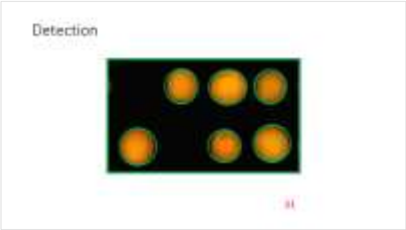
- Dr. Mohamed Azzam

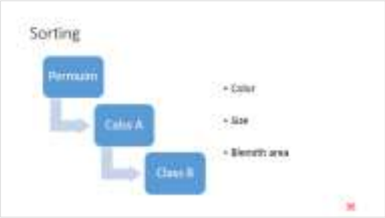


Outline

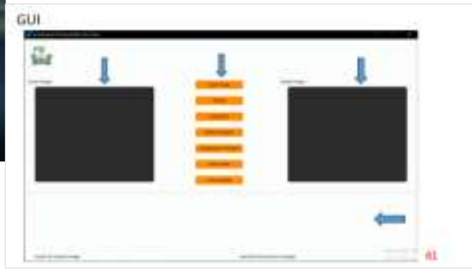


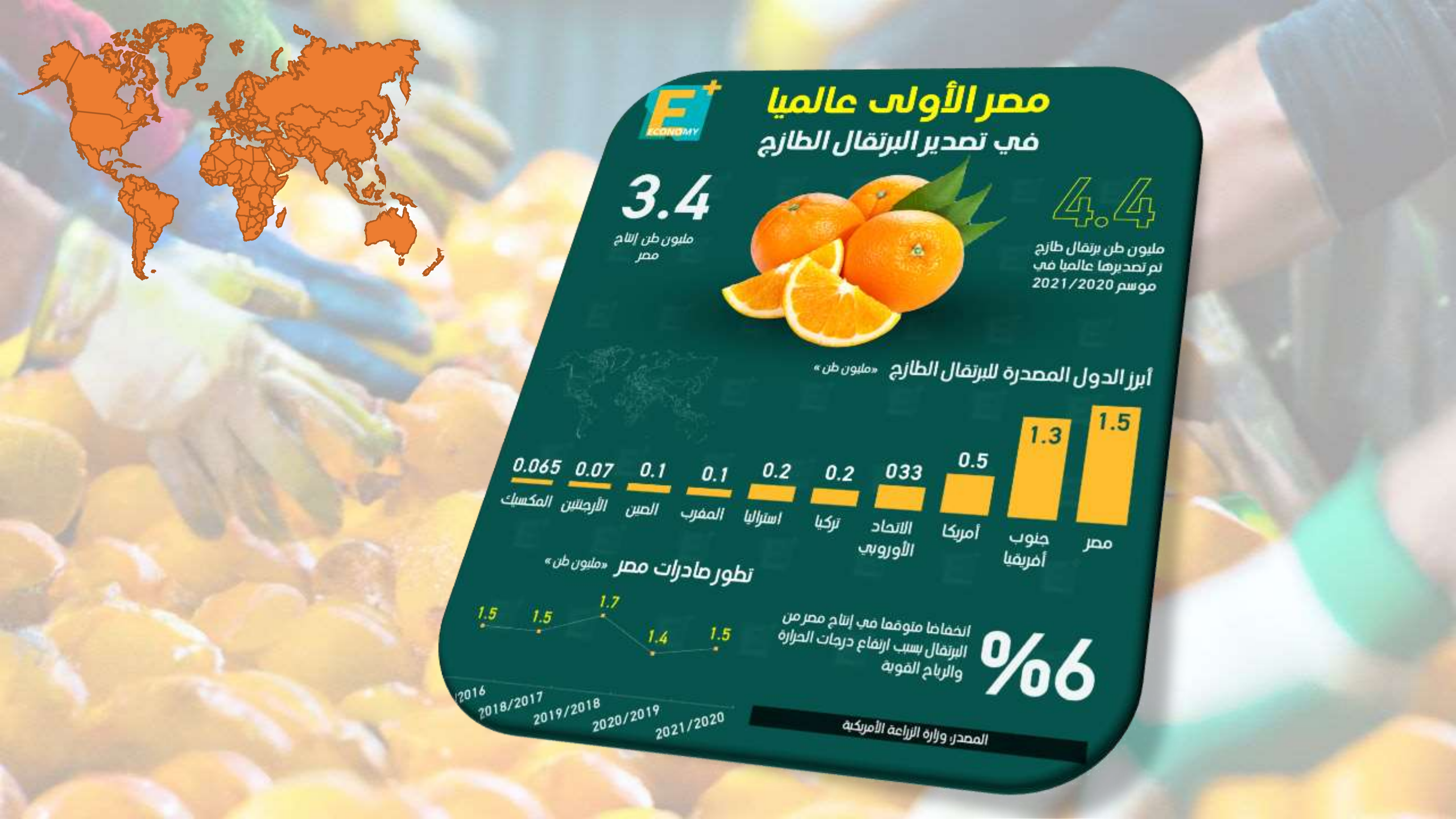

Background
Removal


Detection


Sorting


Defect Detection
Semantic
Segmentation
Thresholding


GUI



مصر الأولى عالميا
في تصدير البرتقال الطازج
3.4
مليون طن إنتاج مصر
4.4
مليون طن برتقال طازج تم تصديرها عالميا في موسم 2021/2020
أبرز الدول المصدرة للبرتقال الطازج «مليون طن»
مصر 1.5
جنوب أفريقيا 1.3
أمريكا 0.5
الاتحاد الأوروبي 033
تركيا 0.2
استراليا 0.2
المغرب 0.1
الصين 0.1
الأرجنتين 0.07
المكسيك 0.065
تطور صادرات مصر «مليون طن»
2016/ 1.5
2018/2017 1.5
2019/2018 1.7
2020/2019 1.4
2021/2020 1.5
%6
انخفاضا متوقعا في إنتاج مصر من البرتقال بسبب ارتفاع درجات الحرارة والرياح القوية
المصدر: وزارة الزراعة الأمريكية

manual sorting

Automatic sorting

Target

Our Process

Capture images

Making enhancements

Detect defects

Classification

Industrial companies that work in this domain

GREEFA

iscansorter

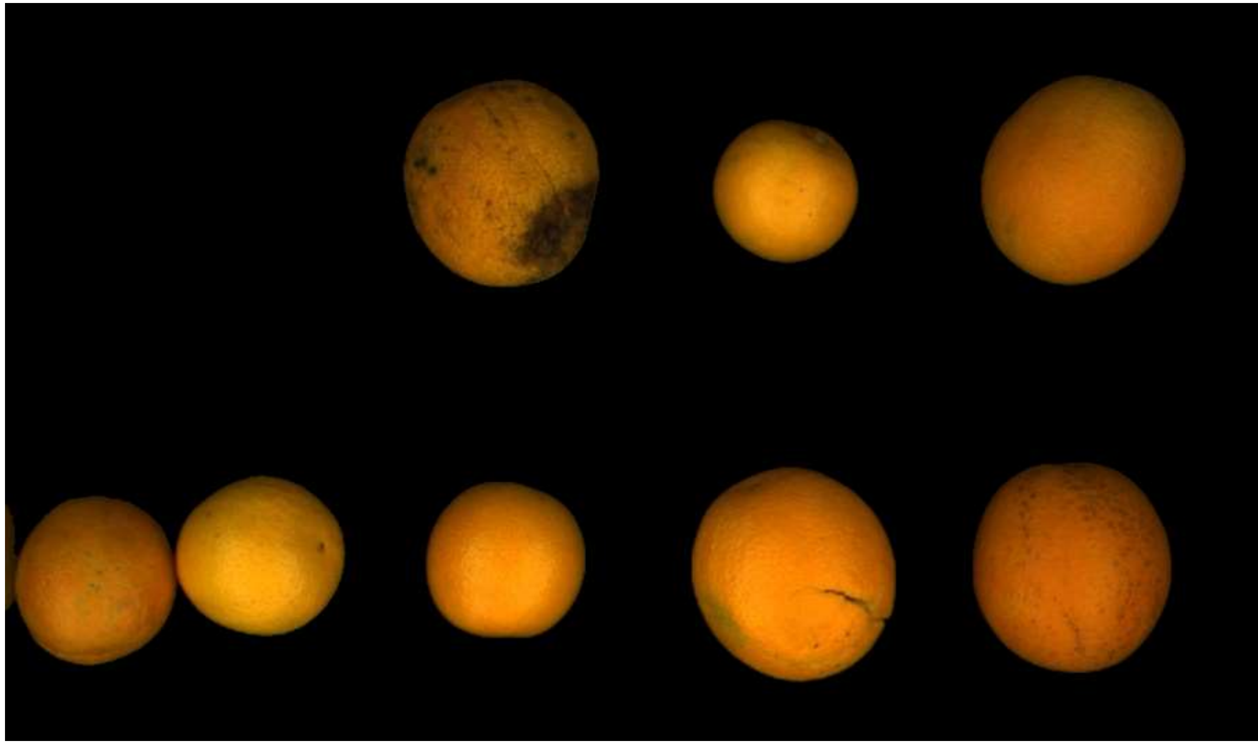
Reemoon

Background Removal

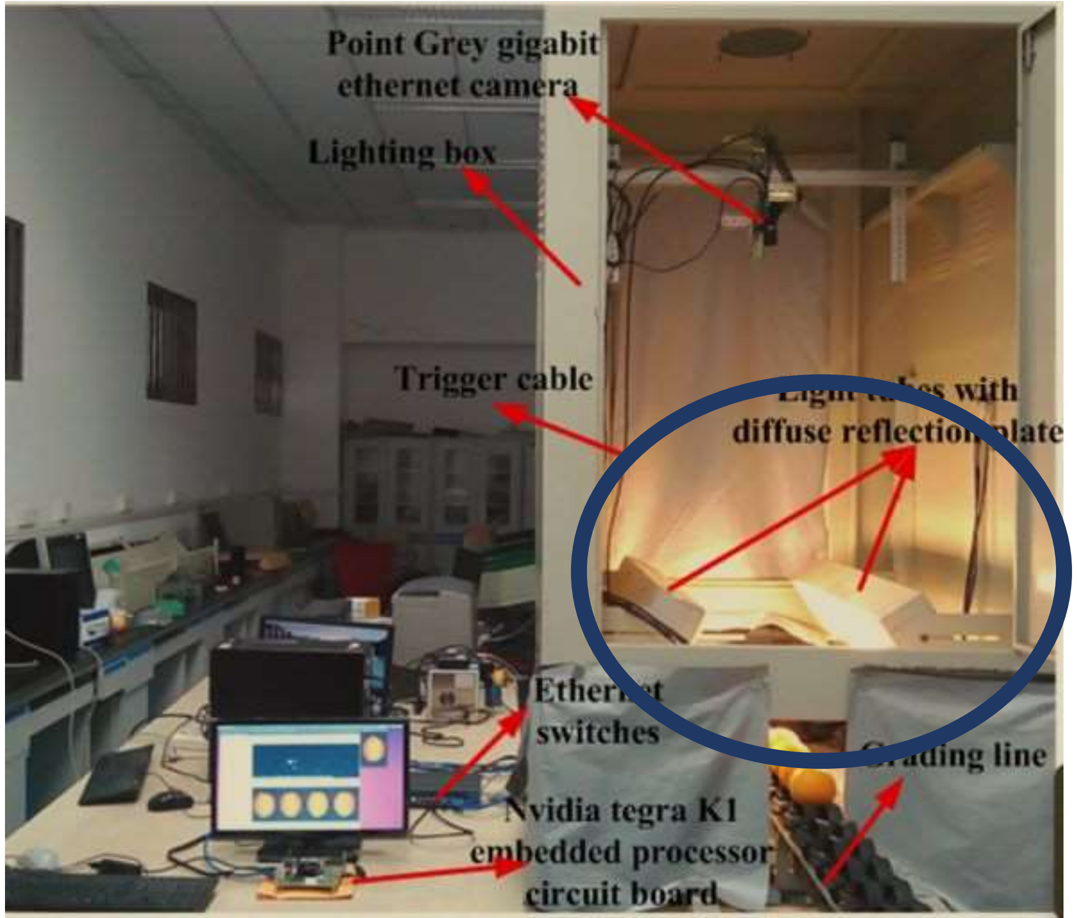
Background Removal: Introduction

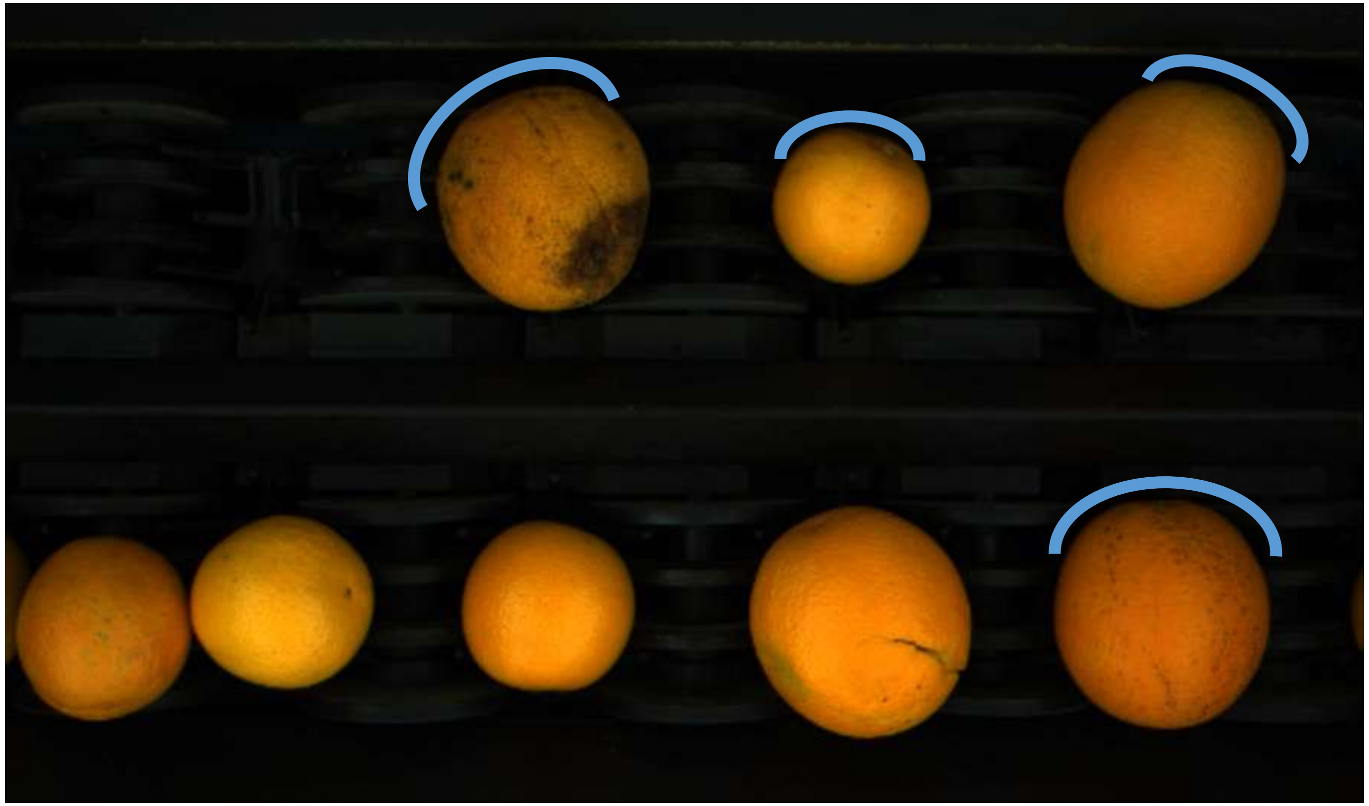
Input image

Result

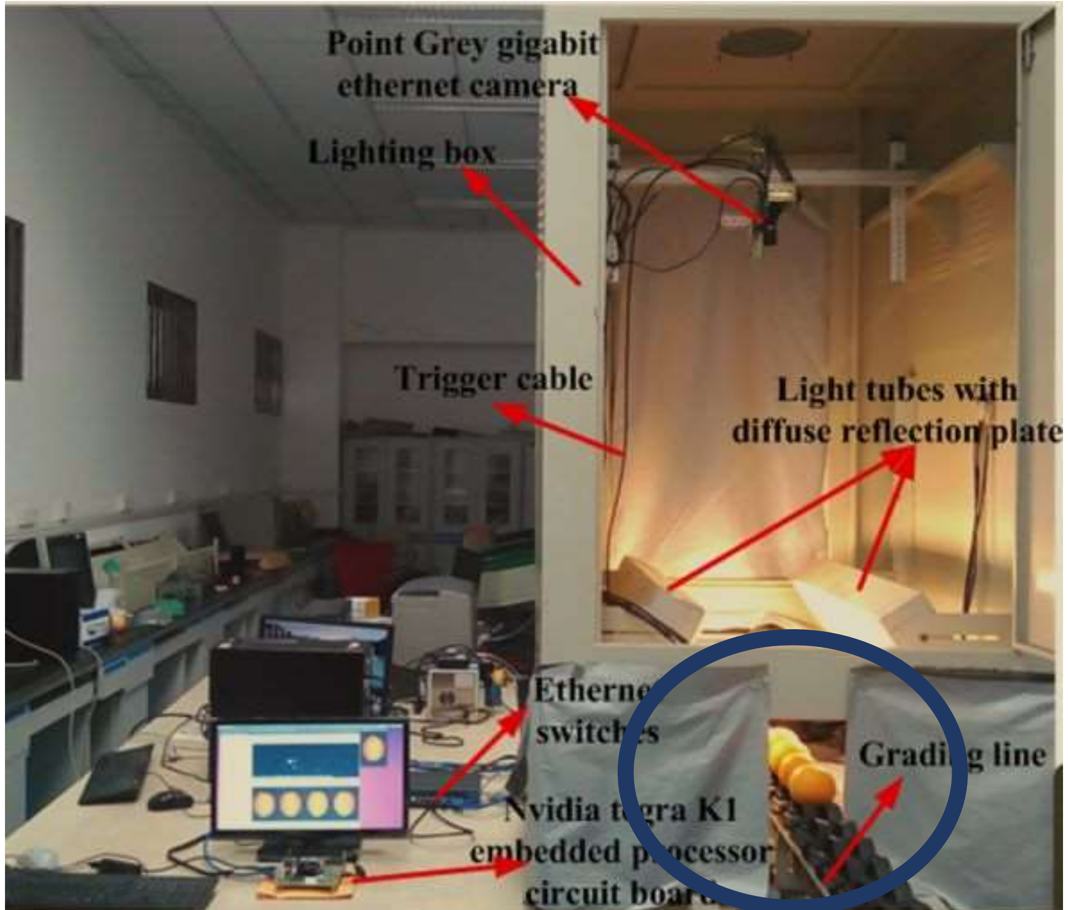


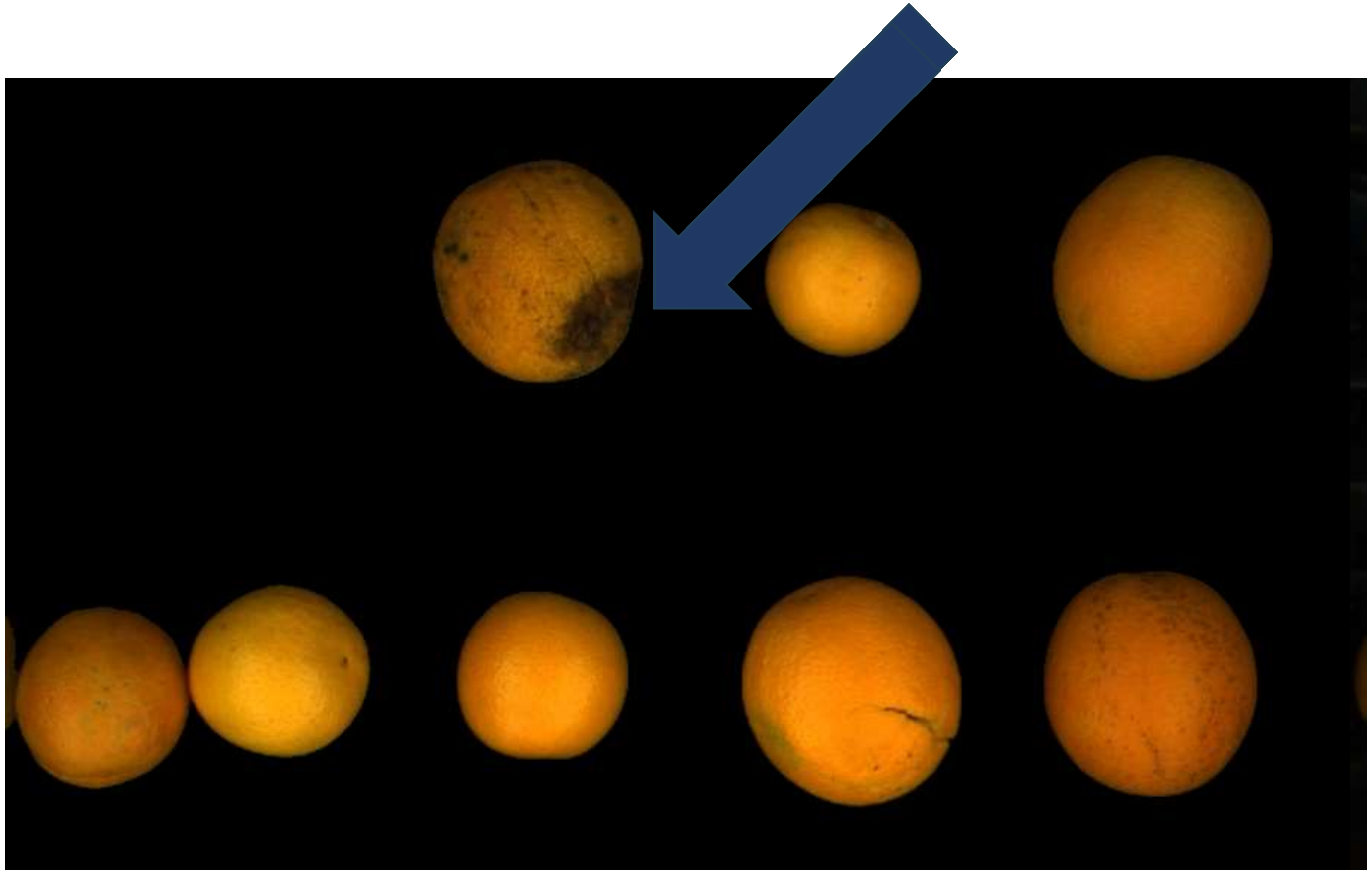
Background Removal: Lightness challenge



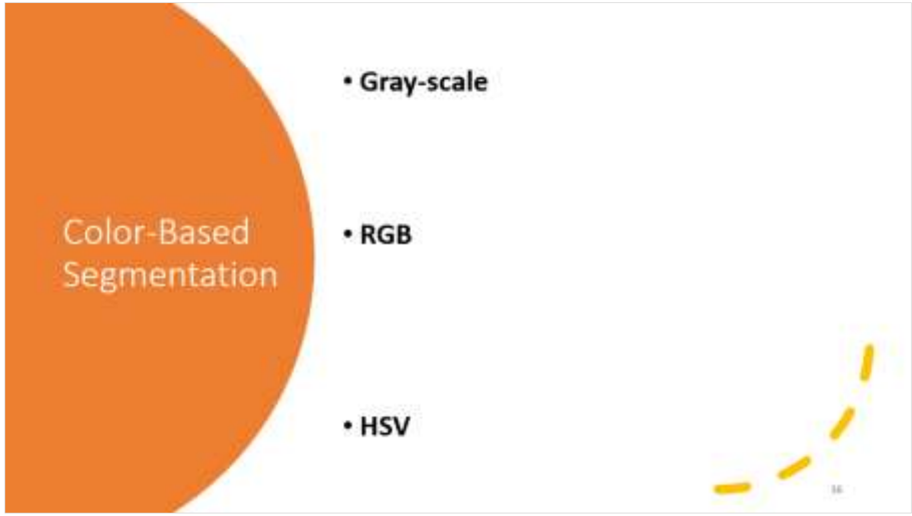


Background Removal: Defect Color





Background Removal: Approaches

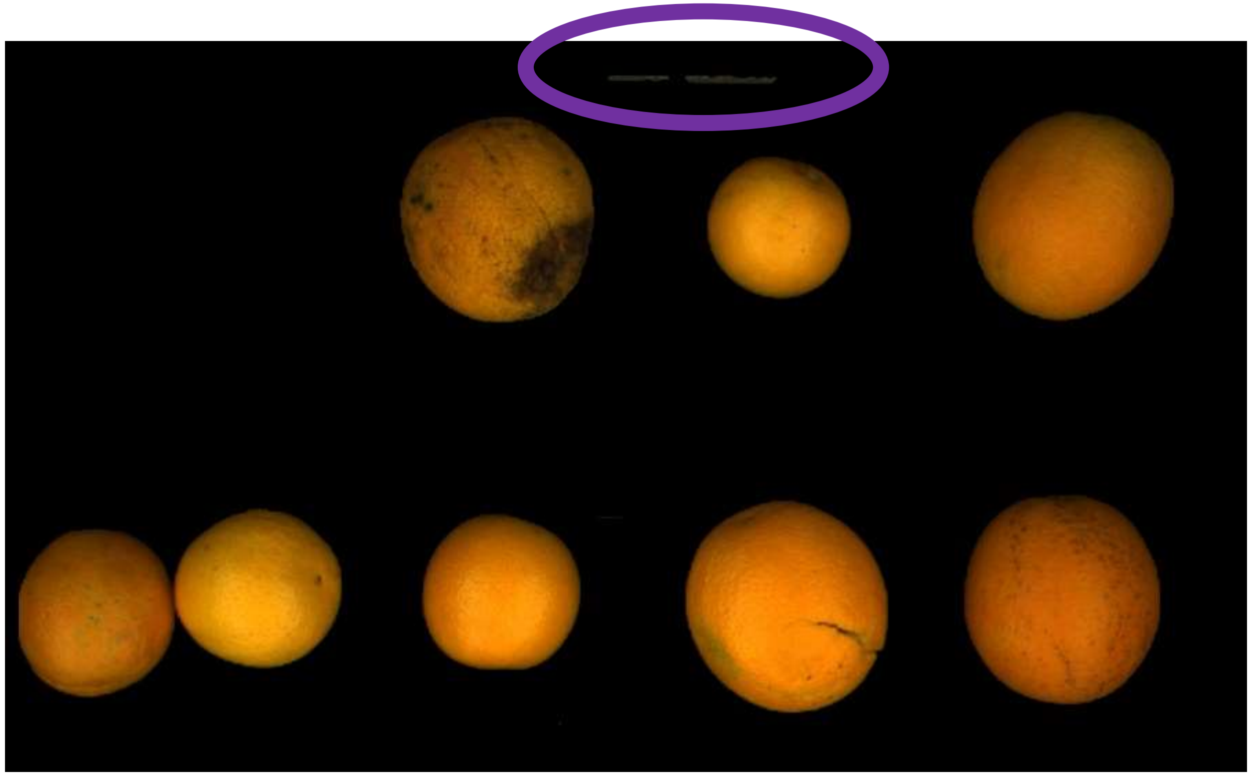


GrabCut Algorithm

GrabCut: Interactive Foreground Extraction

Input image

Result



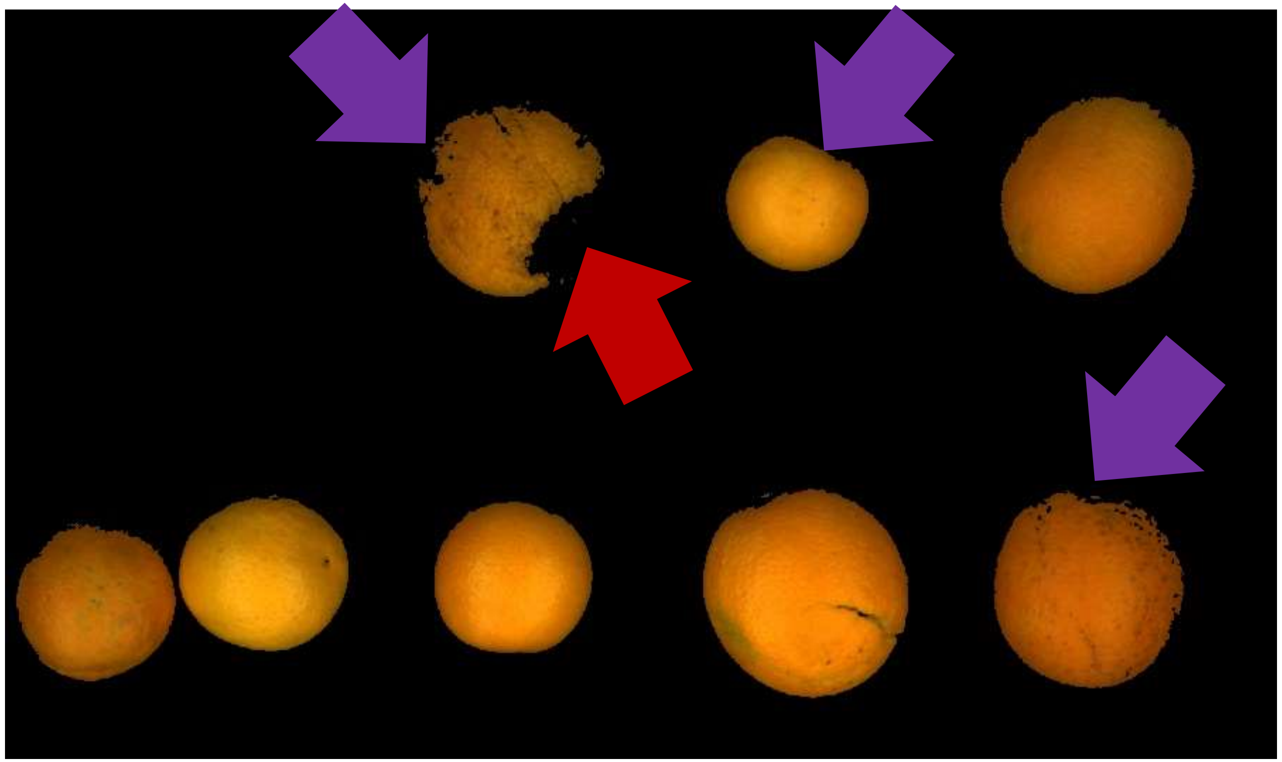
Color-Based Segmentation

- **Gray-scale**
- **RGB**
- **HSV**

Color-based Segmentation: Gray-scale

Input image

Result

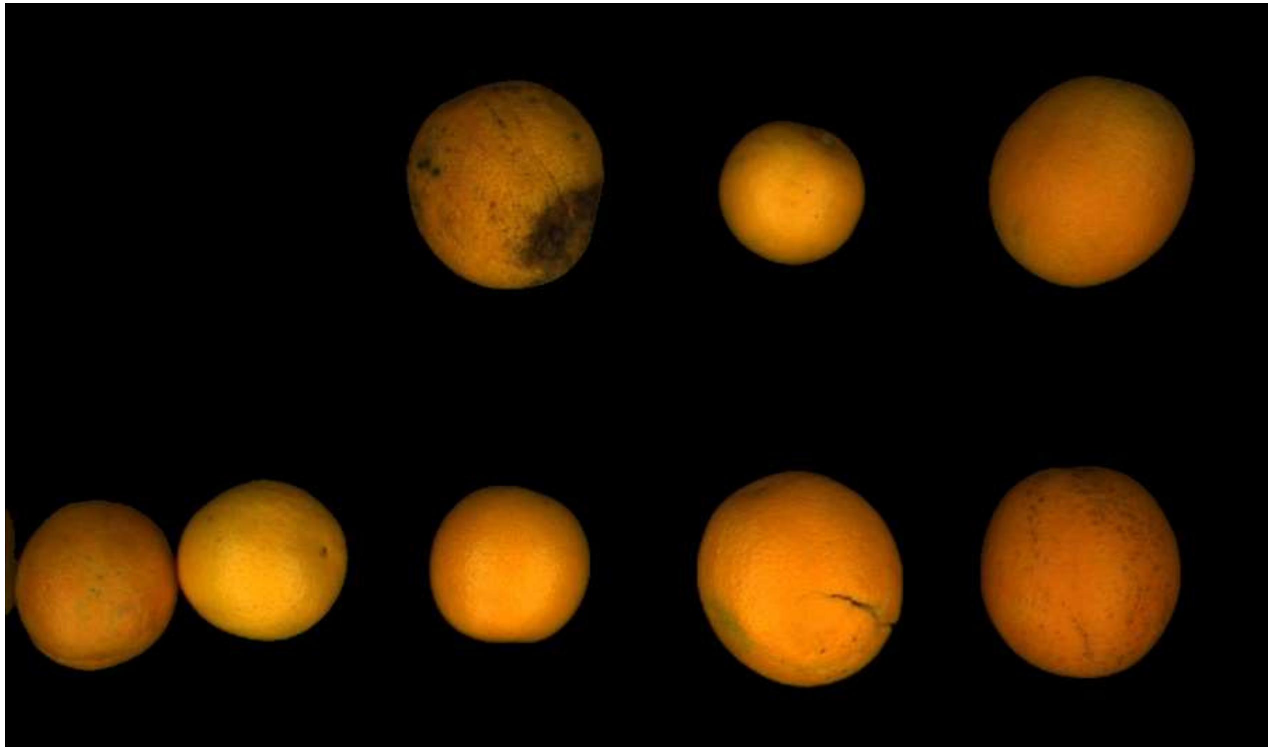


Color-based Segmentation: RGB

Color-based Segmentation: HSV

Input image

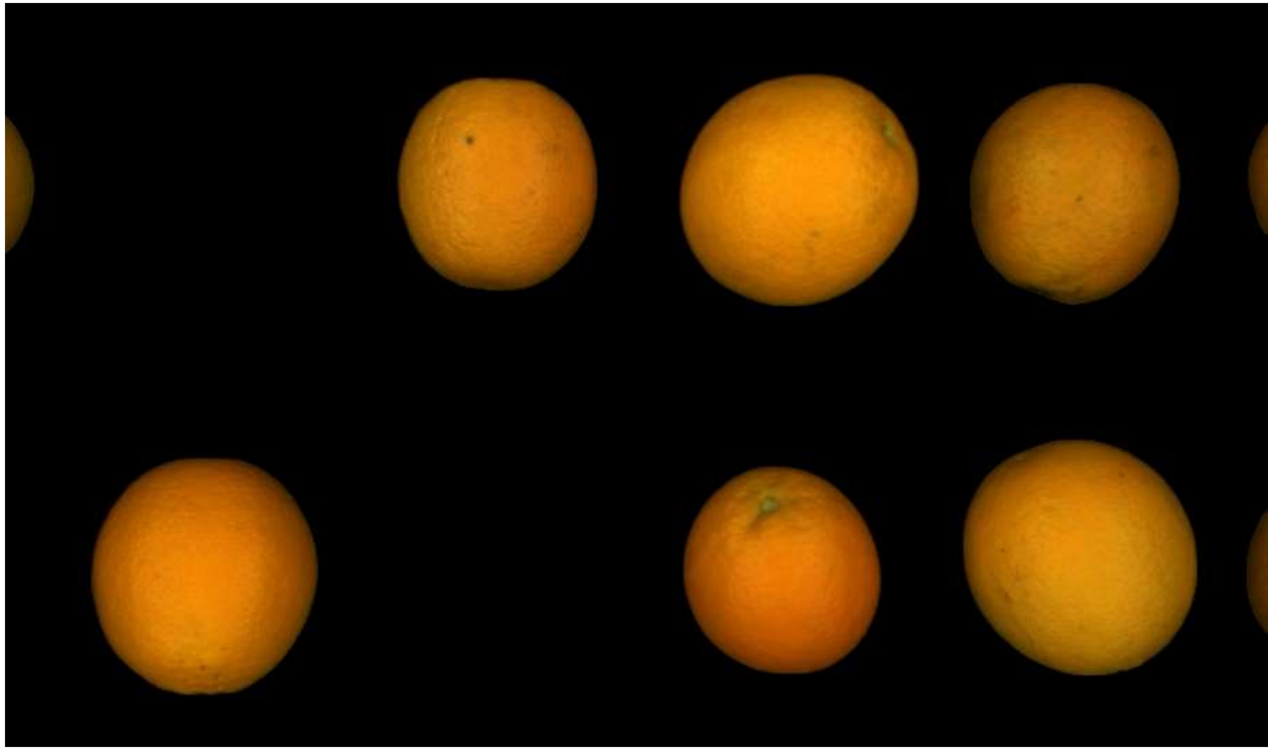
result



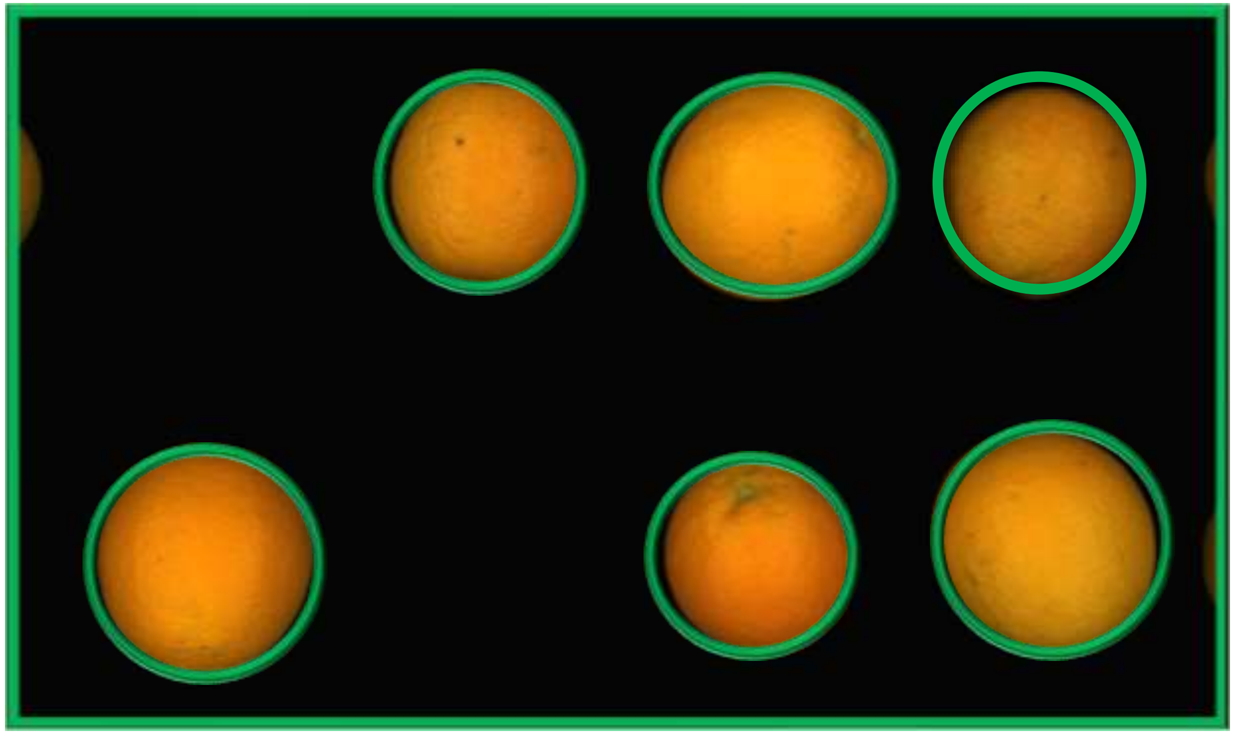
Color-based Segmentation: HSV

Input image

result



Detection



Detection Ways

- Hough Circle Transform
- HSV Space Detection
- Circularity and Ellipticity

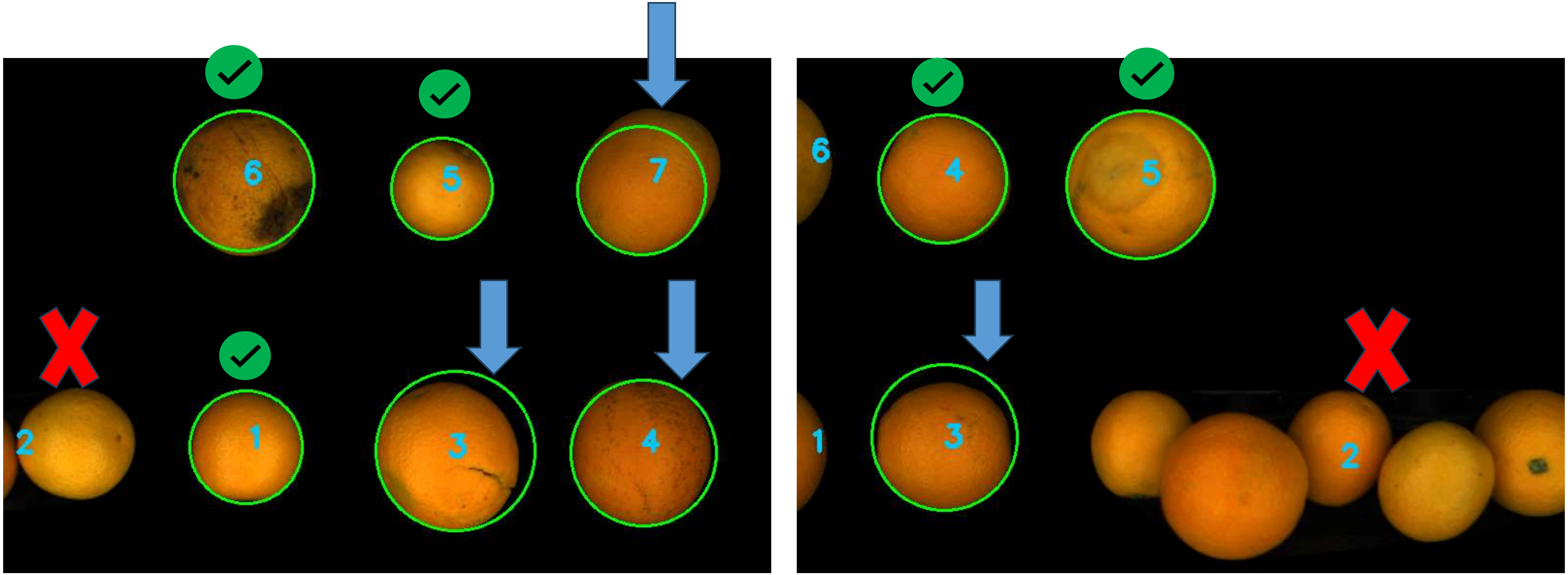
Hough Circle Transform Main idea

- Converting the image from a Cartesian coordinate system to a polar coordinate system

- Then searches for circles by identifying patterns.

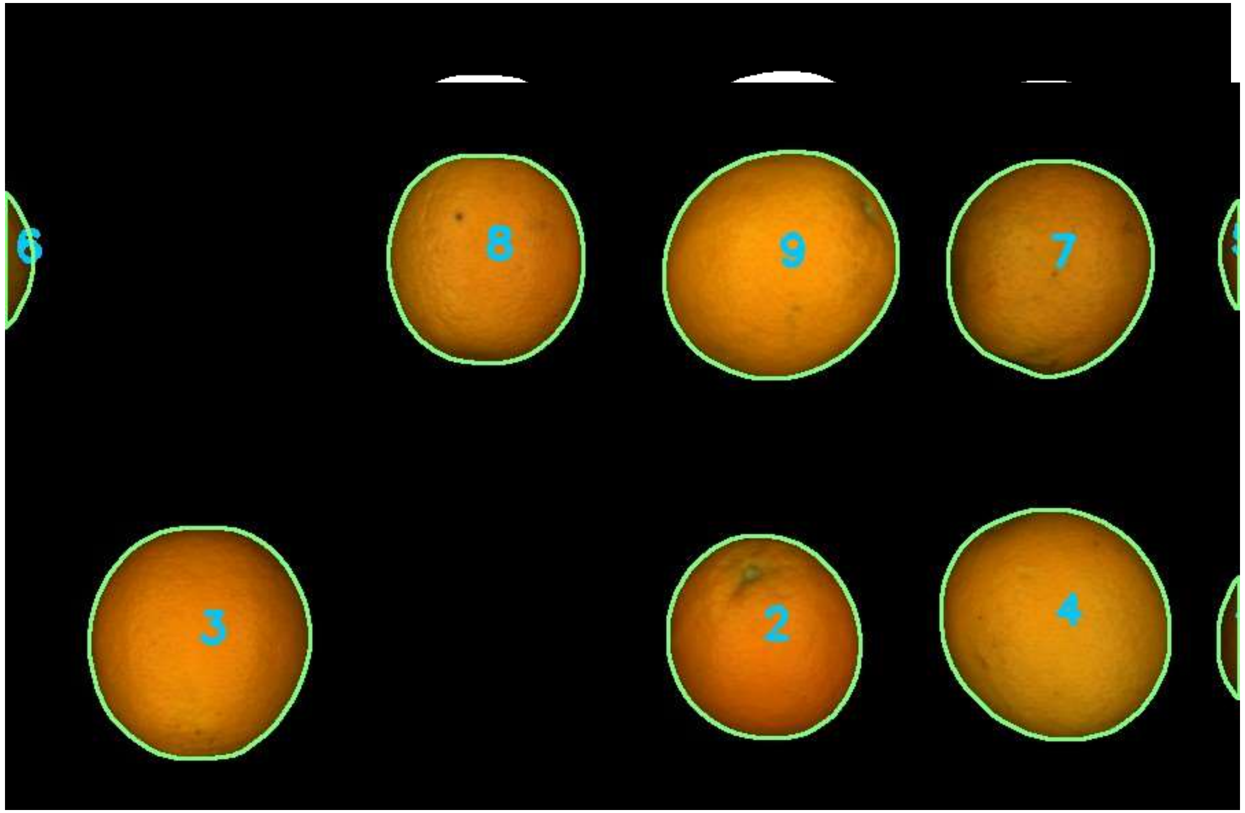
- Each edge pixel in the image accumulates votes for possible circle centers.

Output



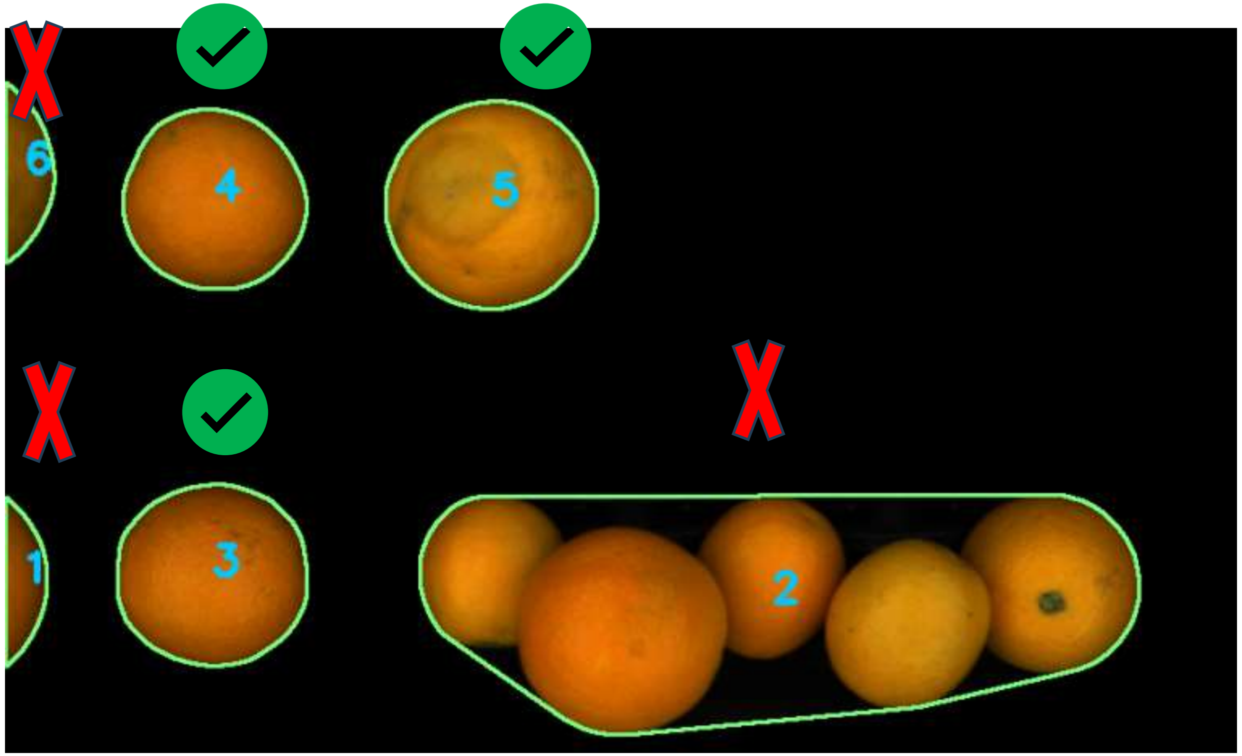
HSV Space Detection

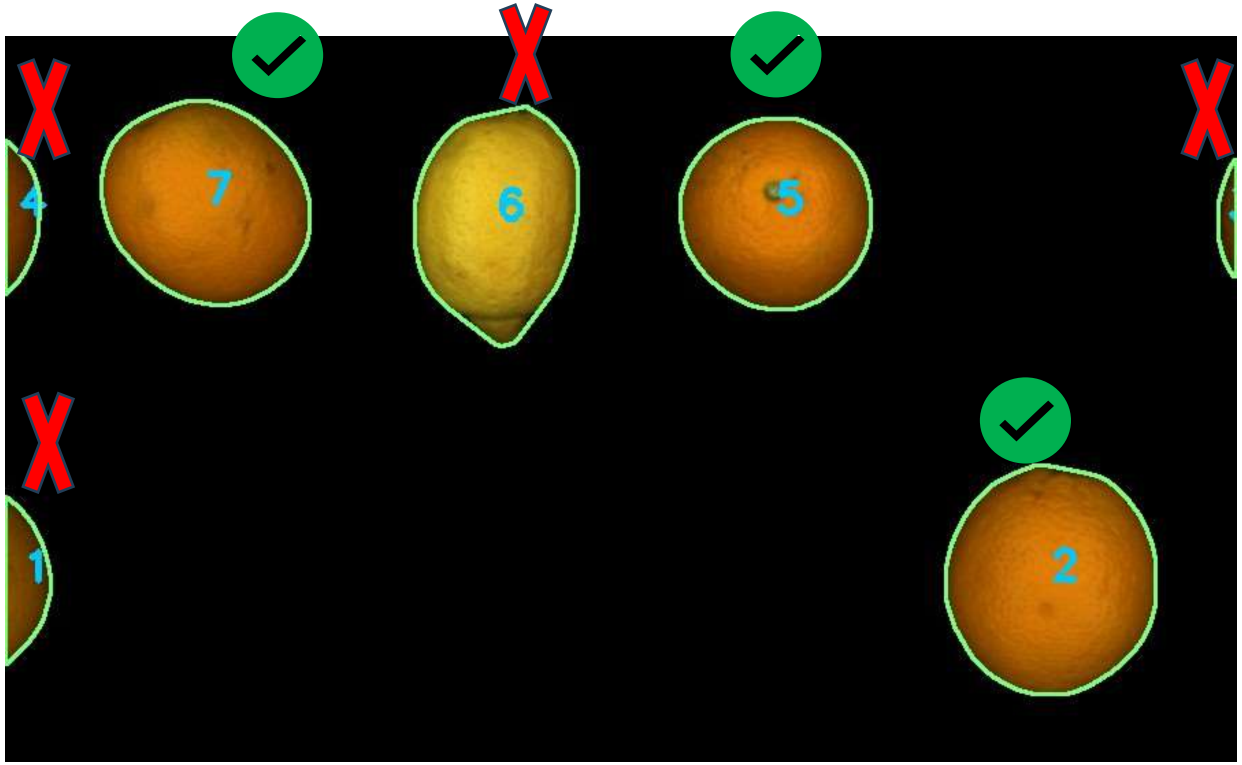
- Take masked image from segmentation
- Search for contours
- Drow contours



Output

- Perfectly detect all contours
- There are no wanted and overlapped detected





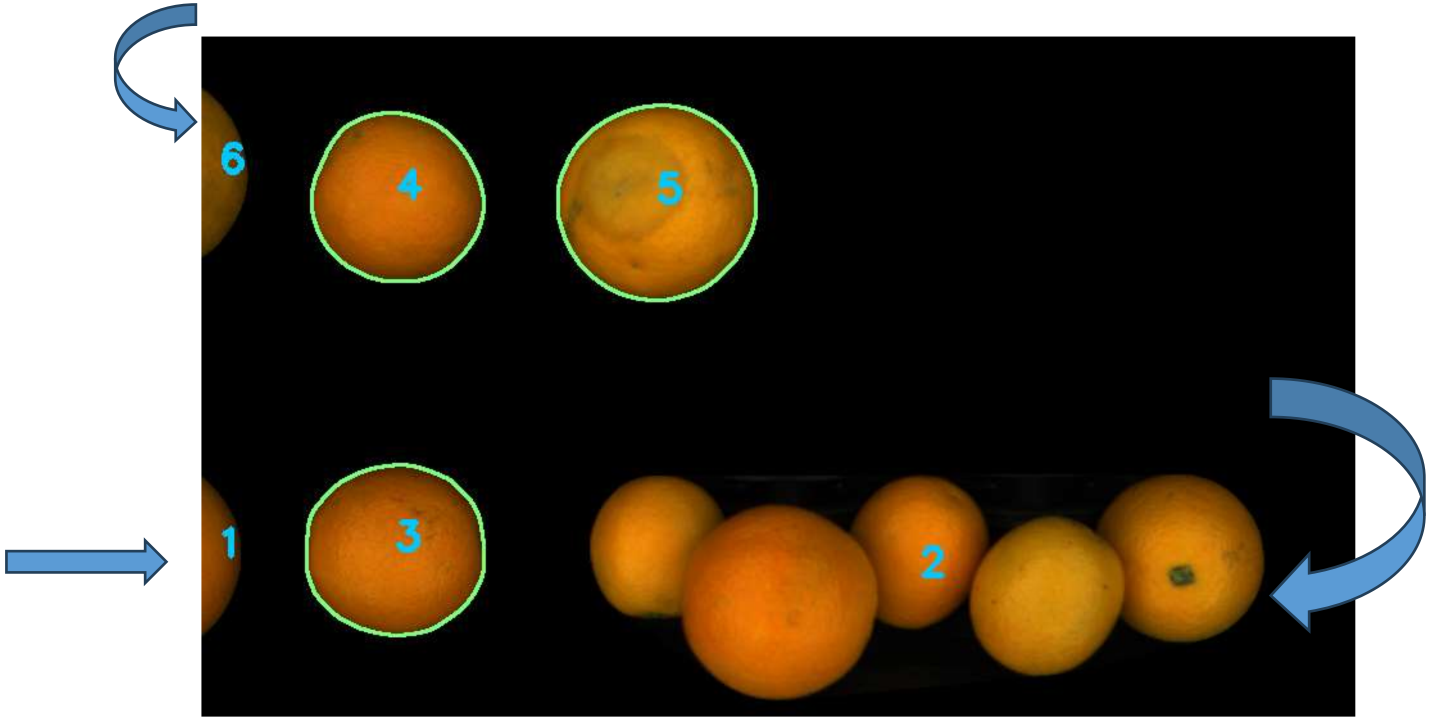
Overlapping

- HSV Area constrain
- Euclidean distance
- Circularity & Ellipticity

HSV Area constrain

- Take Contour in the range 5000 to 25000

- Solve small and
  big contours

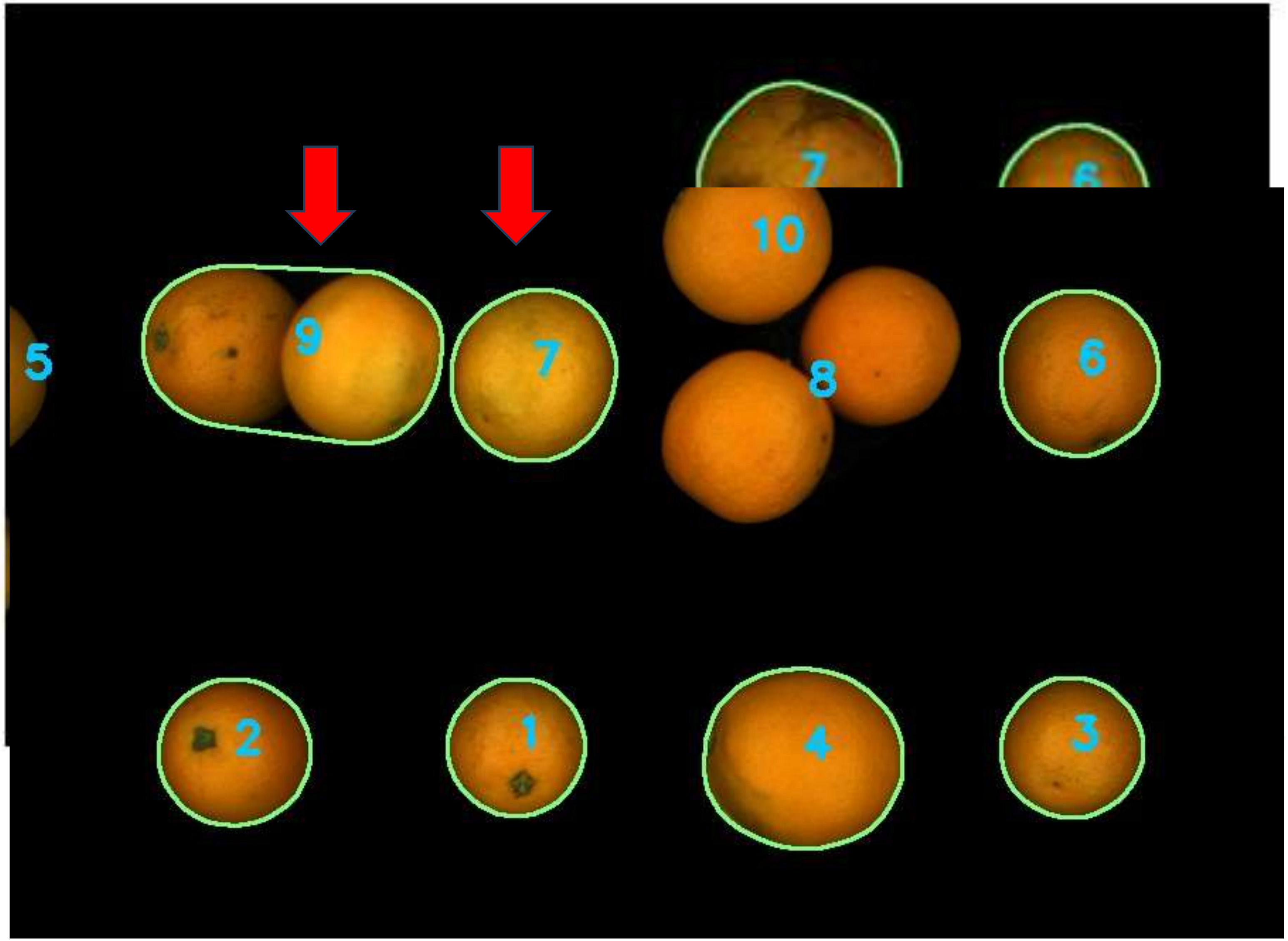


HSV Area constrain Problems

- Overlapped

Contour < 25000

- Near contours

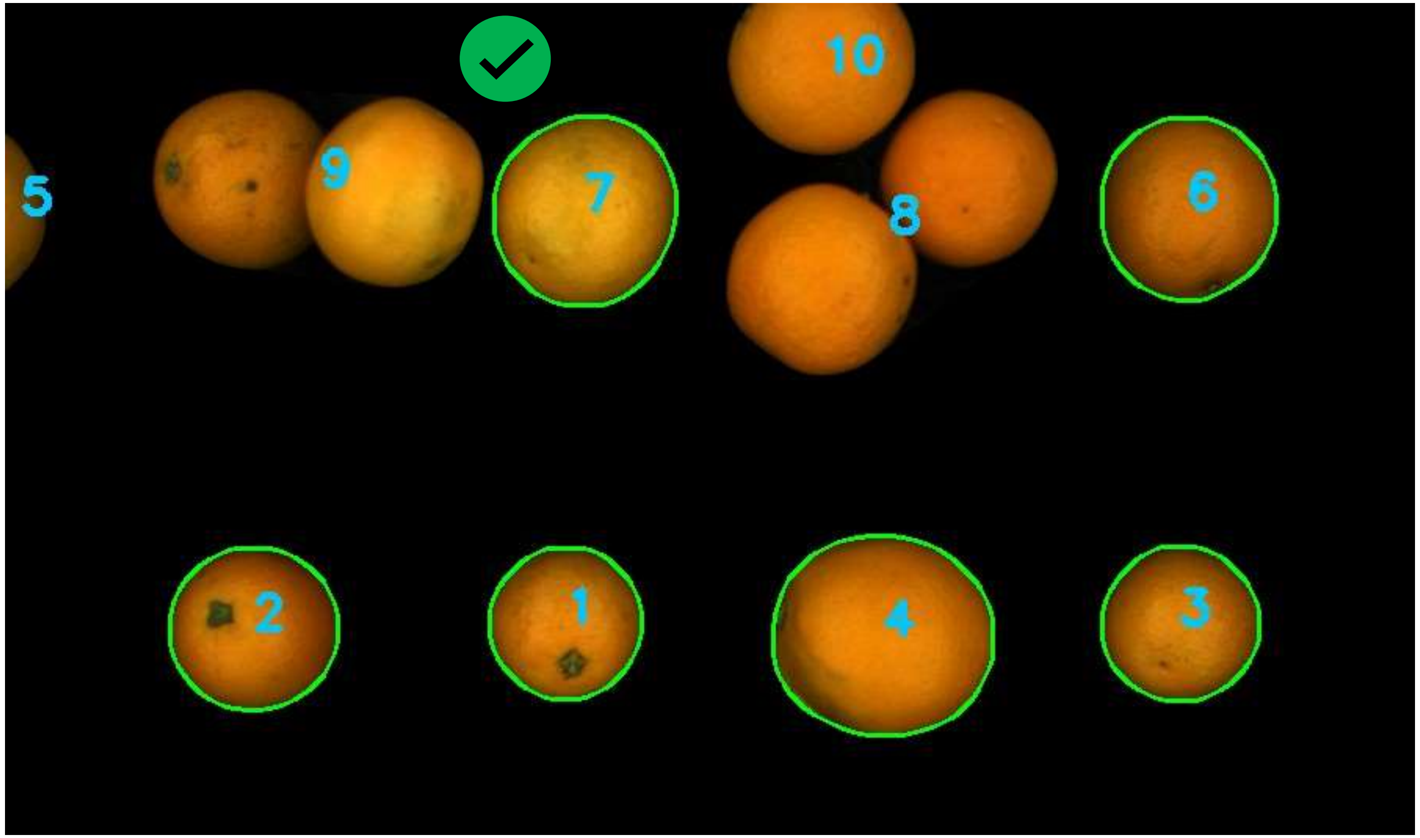


Euclidean distance

- Claculttate the distance between the two centers
- $d = \sqrt{((x2 - x1)^2 + (y2 - y1)^2)}$.
- Ignor the nearest centers

Output

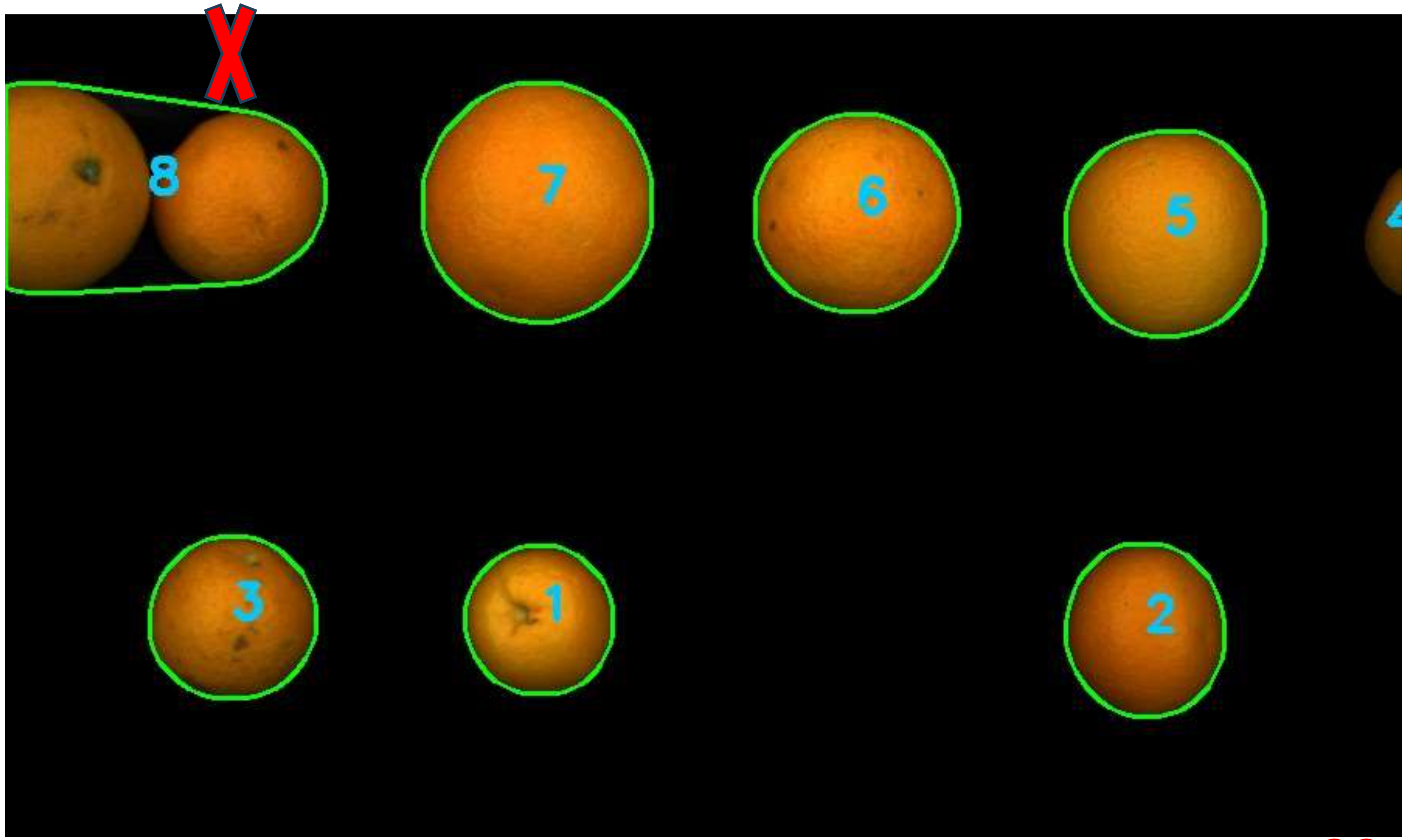
- Solve Near contours



Problems

- Overlapped

Contour < 25000

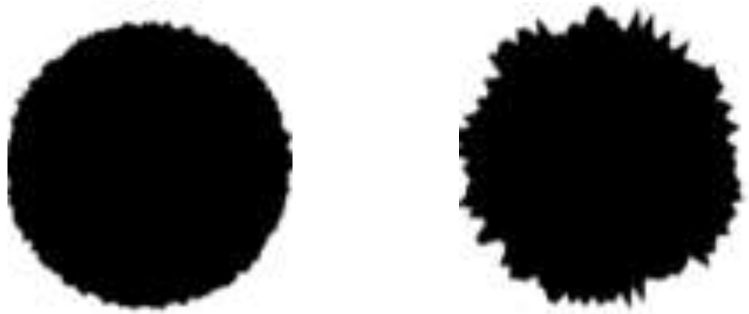


Circularity & Ellipticity

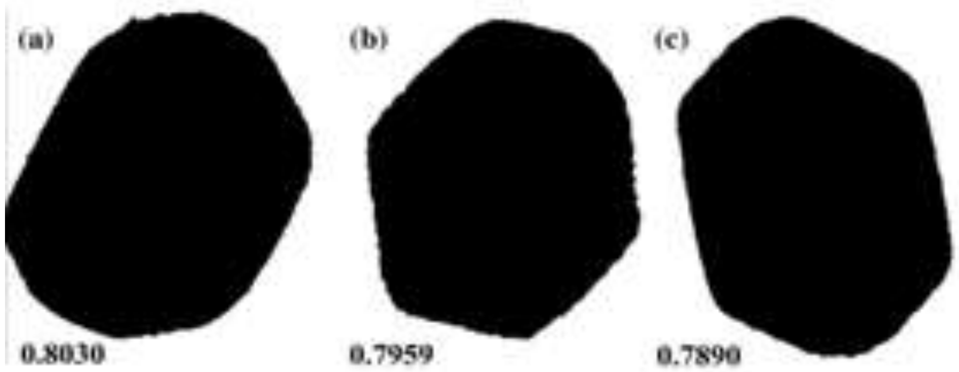
- Circularity = $\dfrac{(4 * \pi * \text{Area})}{(\text{Perimeter}^\wedge 2)}$

  Therthold 95

- Ellipticity = $\dfrac{\text{MajorAxisLength}}{\text{MinorAxisLength}}$
  - Therthold 95

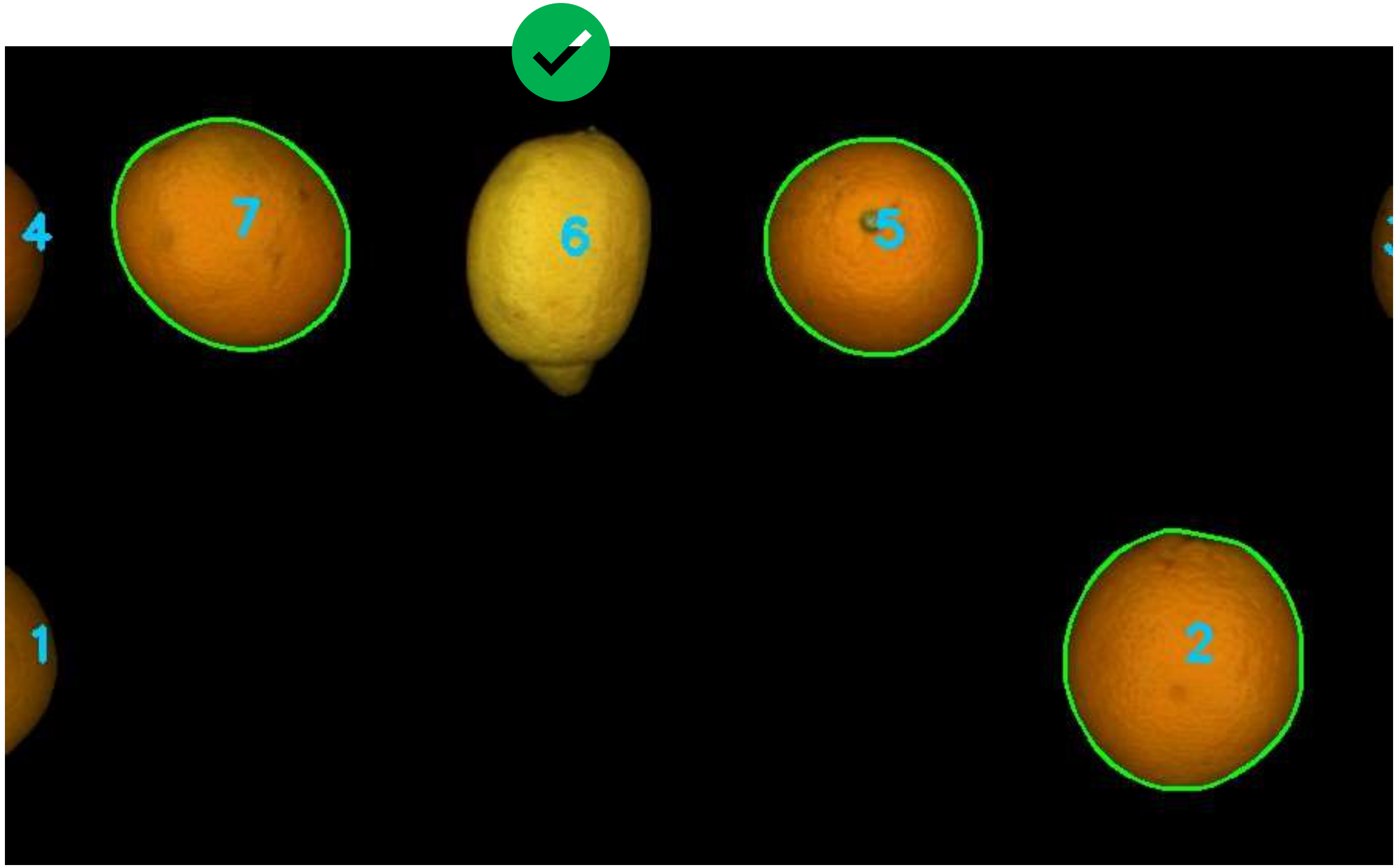


(b) 0.989 (c) 0.965



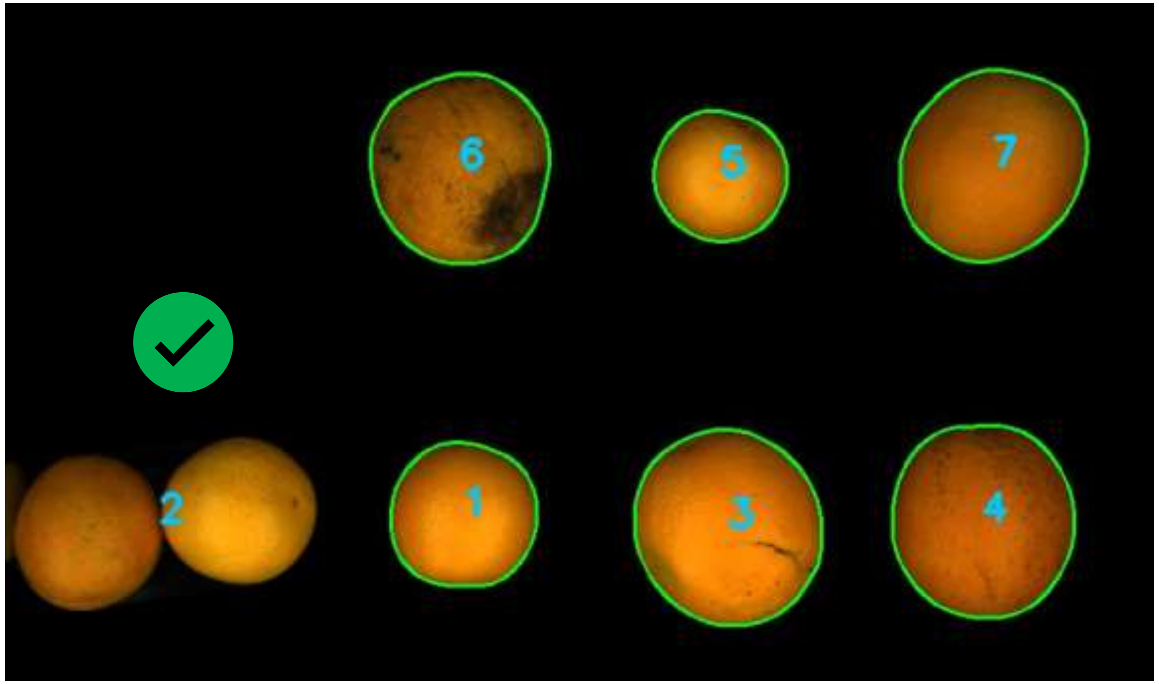
Output

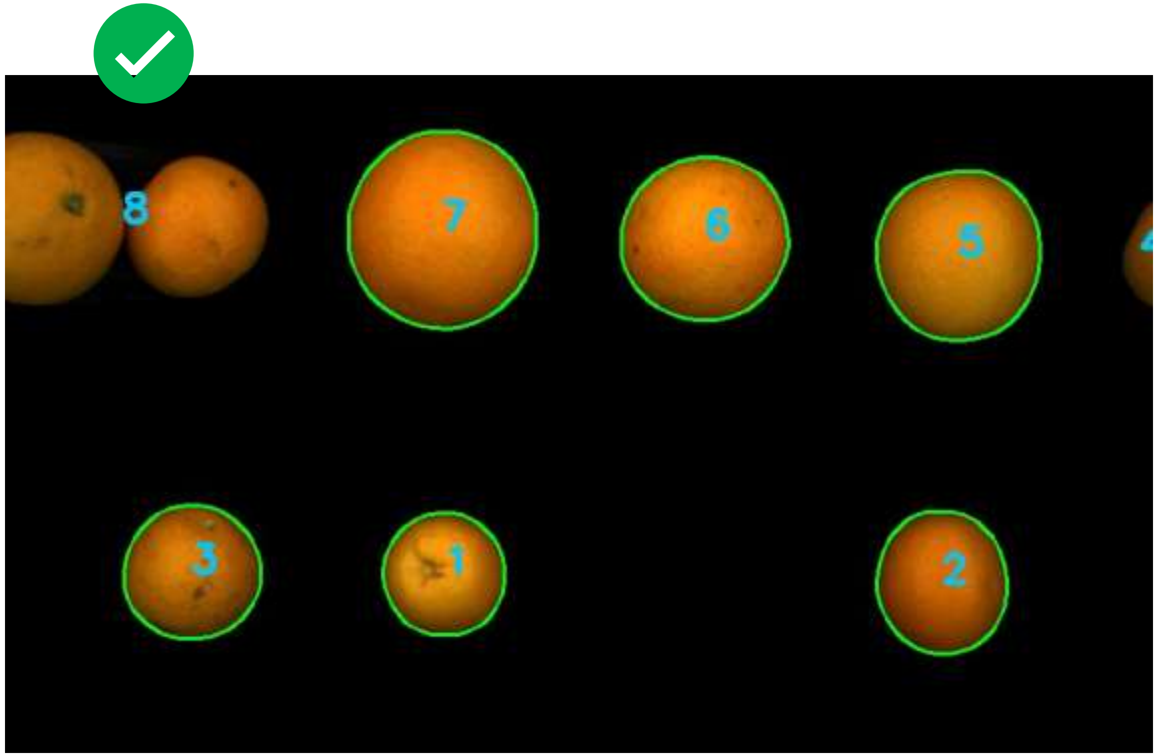
- Ignore anting not orange



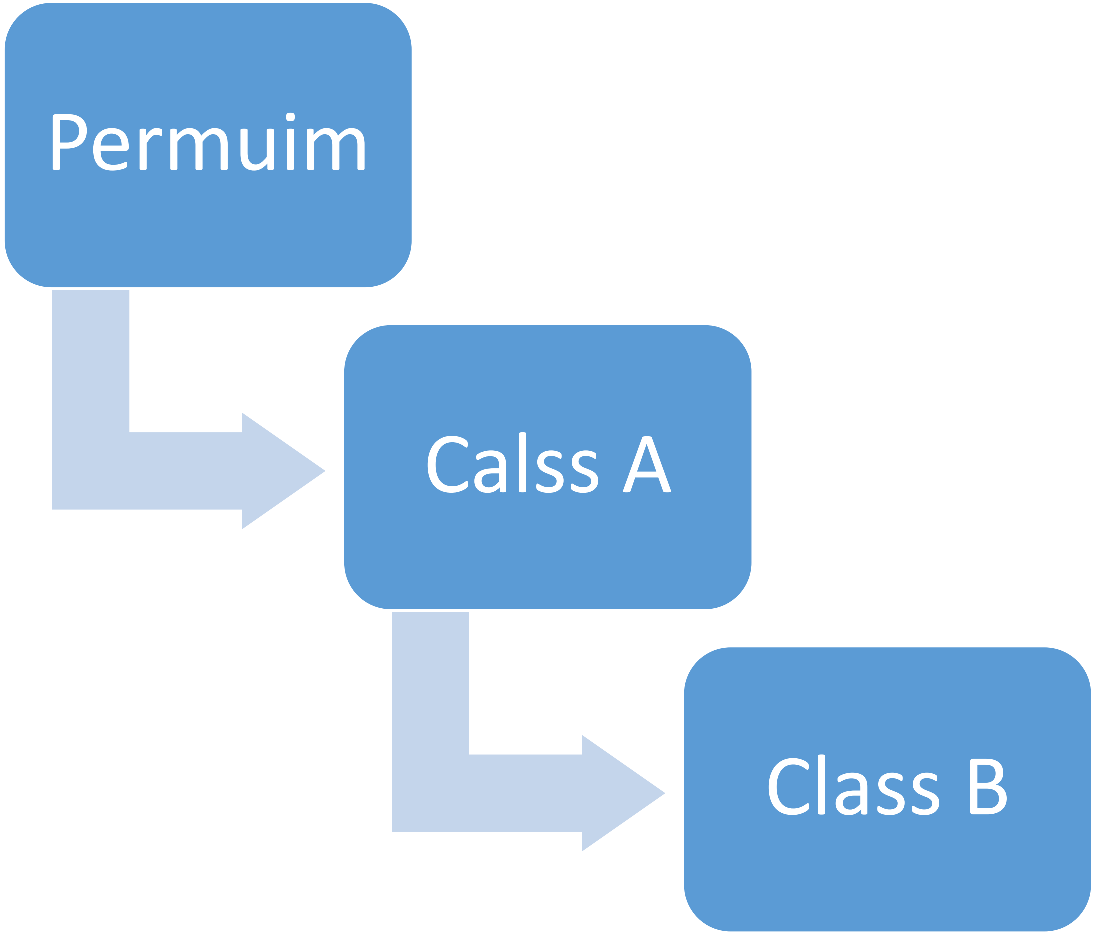
Output

- Solva Overlapped Contour < 25000





Sorting



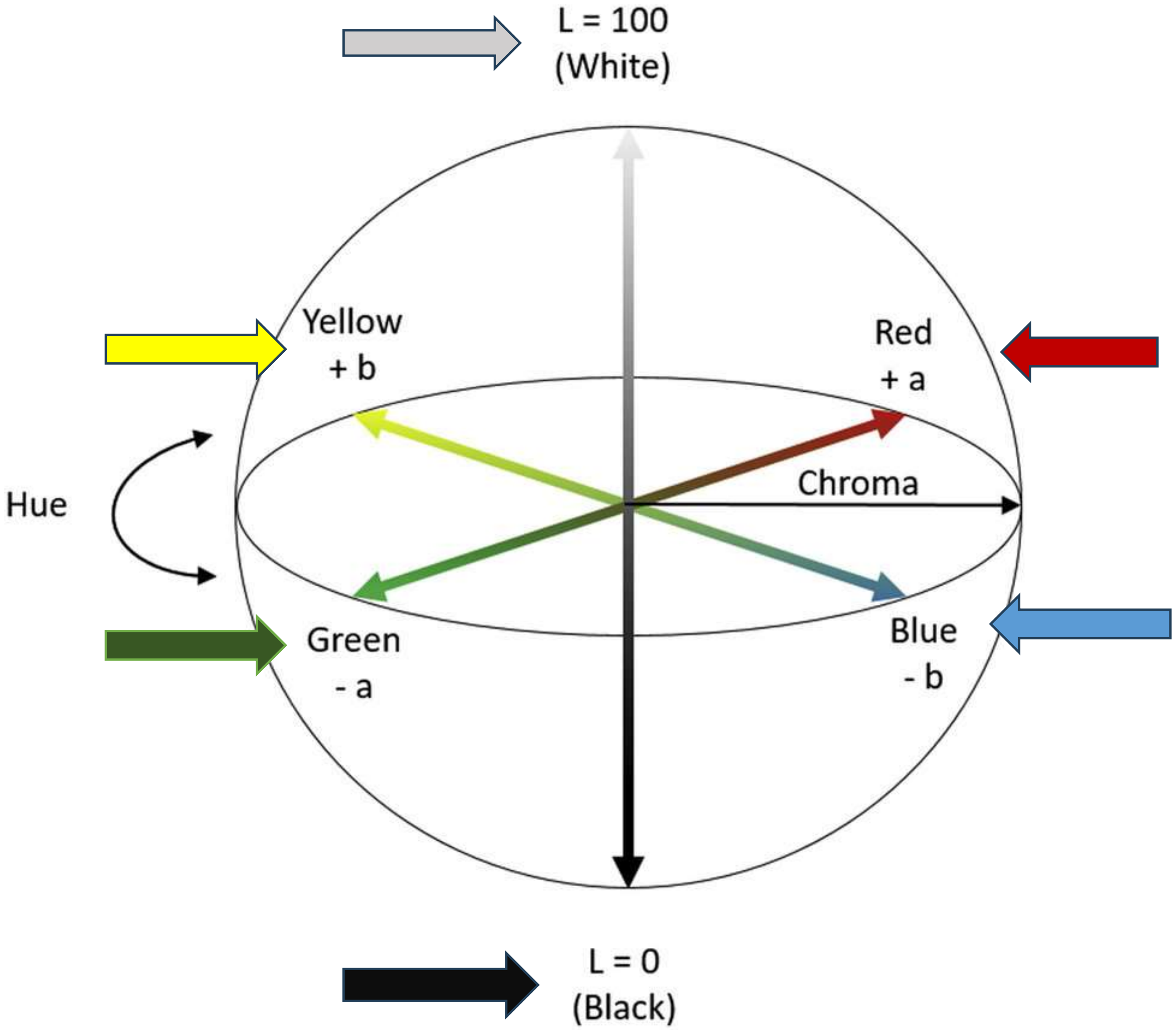
- Color
- Size
- Blemith area

Citrus Color Index

- CCI= $\frac{100 * a}{L * b}$

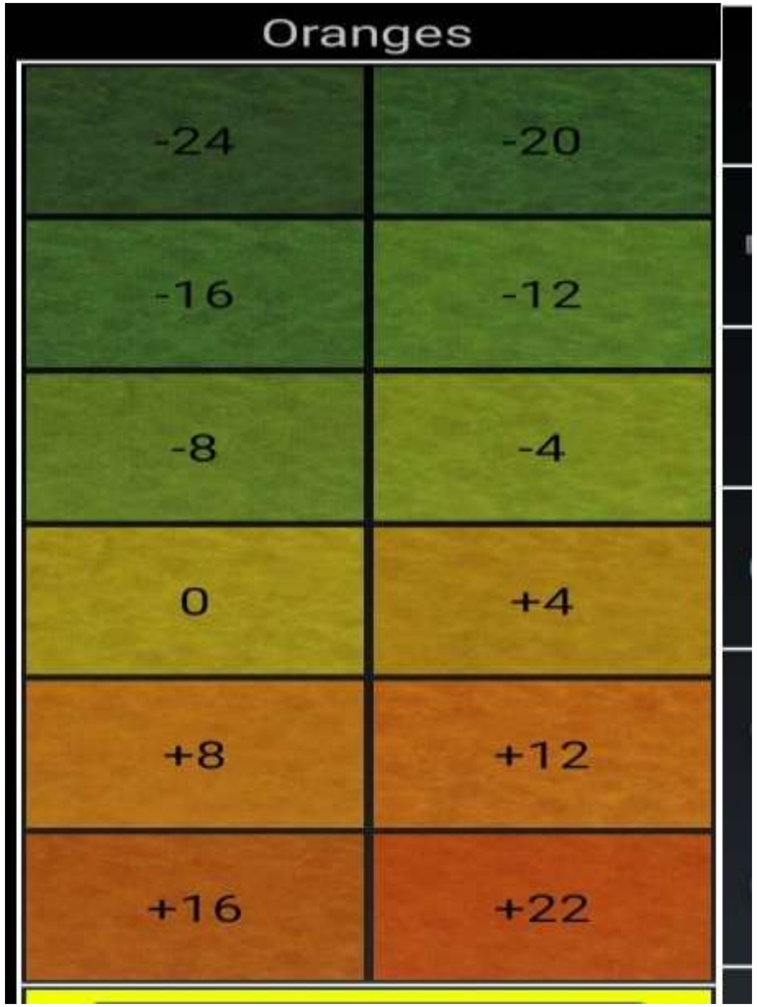
- LAB Space

- L : Reference illuminant
- a : mean value of channel a
- b : mean value of channel b

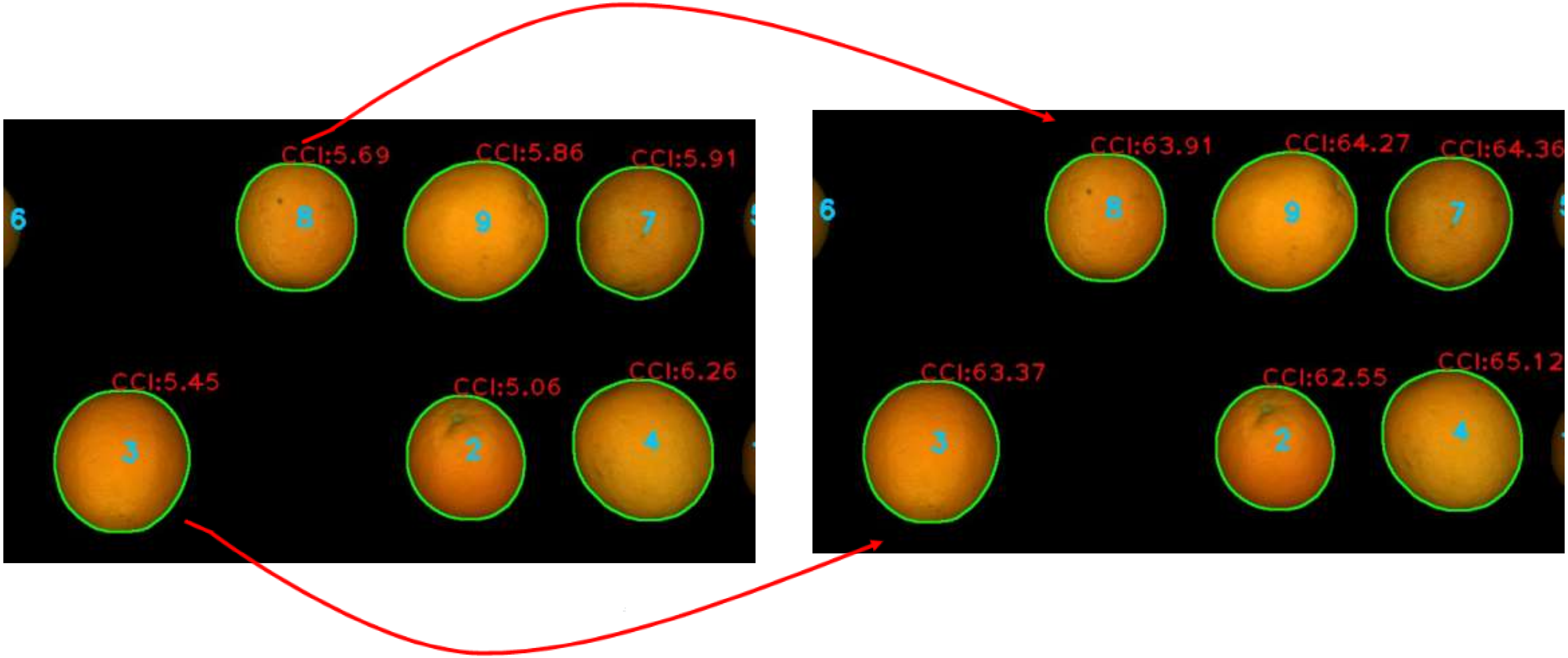


Color index

- CCI Range: -22 –> 24
- Most orange orange
- Mapping to 0–>99

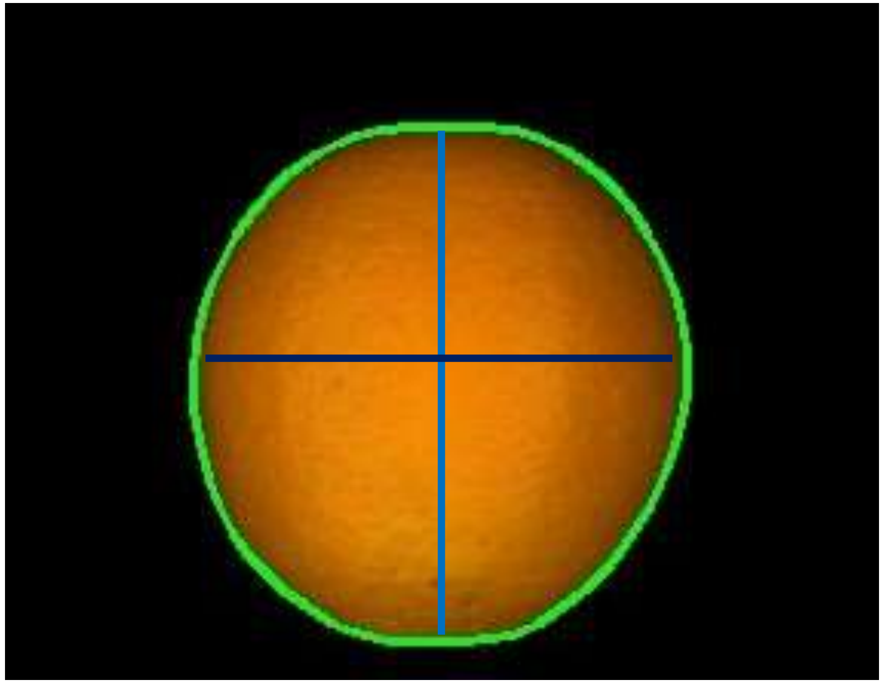


Mapped CCI


CCI:5.69
CCI:5.86
CCI:5.91
6
8
9
7
CCI:5.45
CCI:5.06
CCI:6.26
3
2
4
CCI:63.91
CCI:64.27
CCI:64.36
6
8
9
7
CCI:63.37
CCI:62.55
CCI:65.12
3
2
4

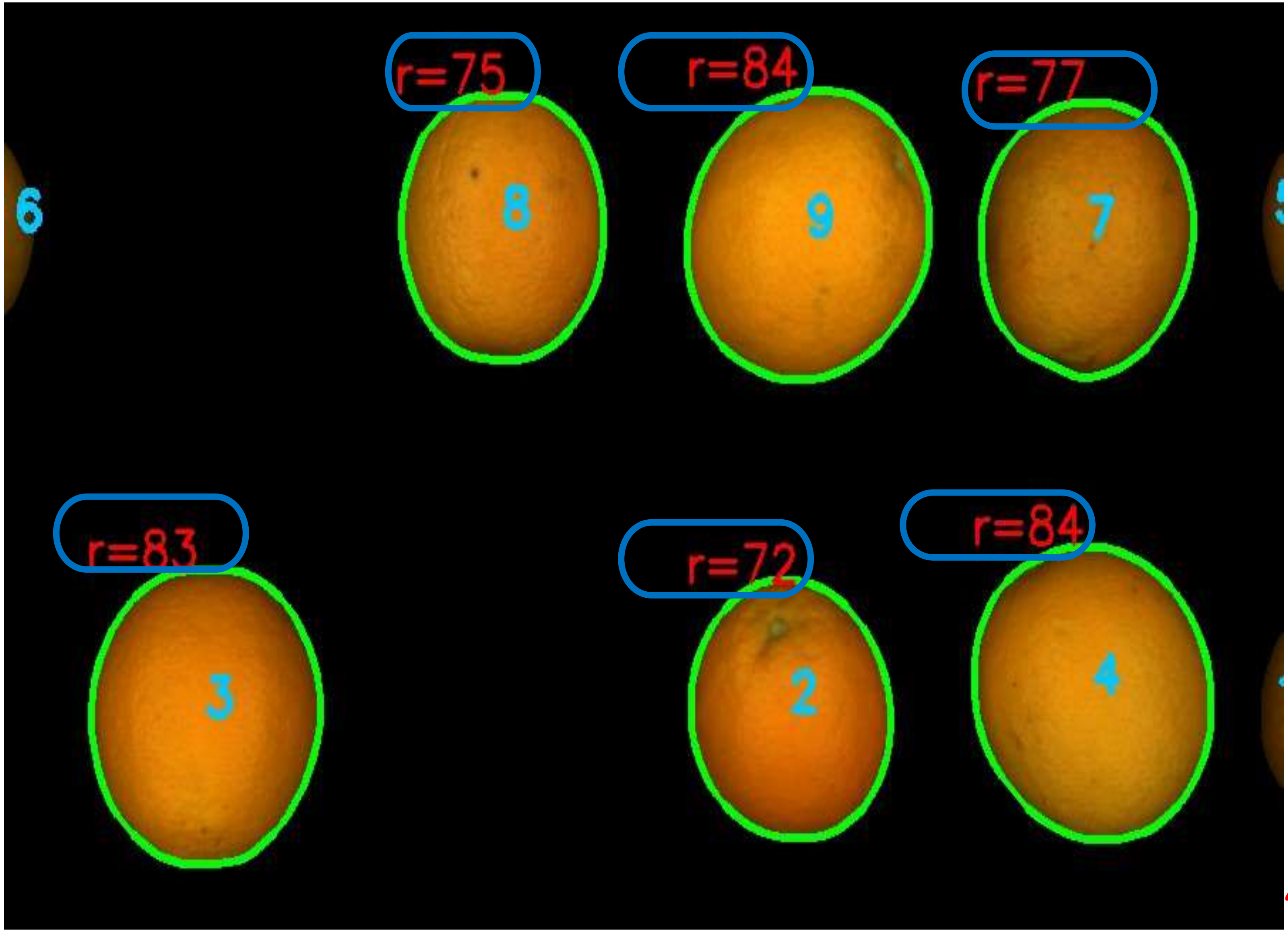
Size (Radius)

$$\text{Radius} = \frac{(\text{major\_axis} + \text{minor\_axis})}{4}$$



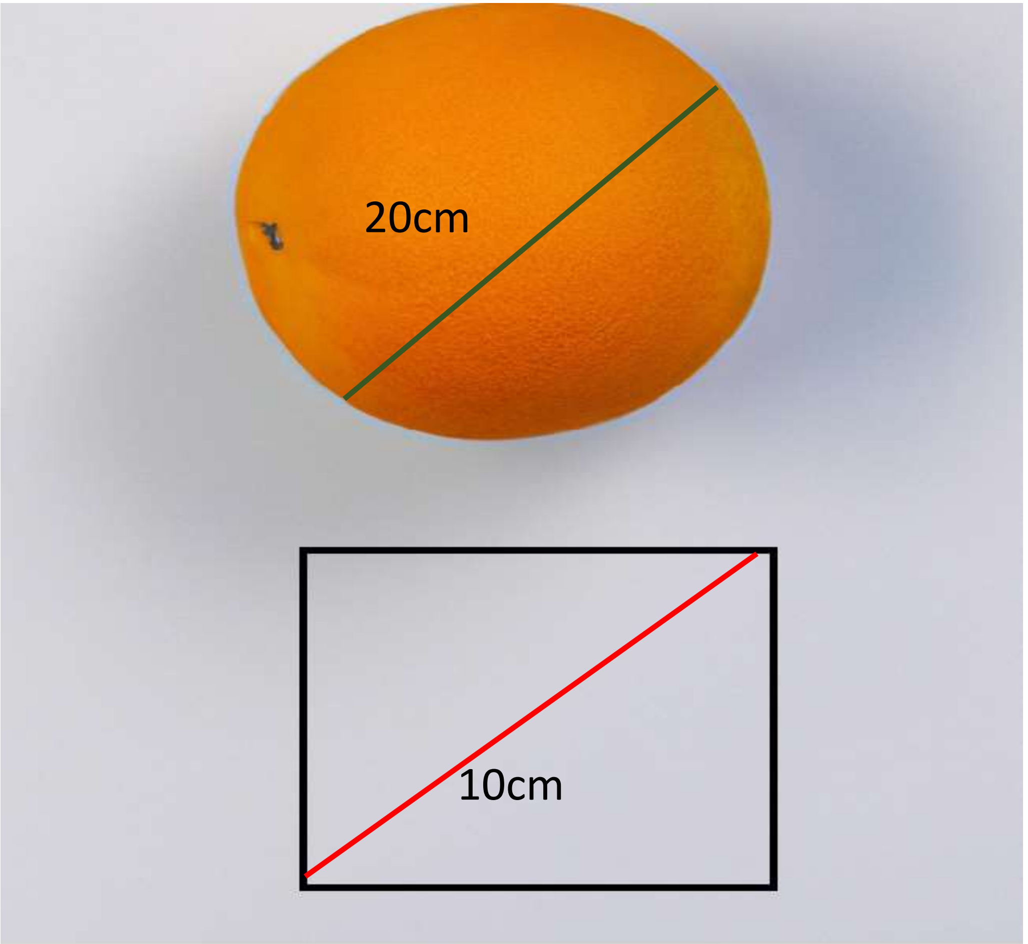
Radius

Depth
per inch

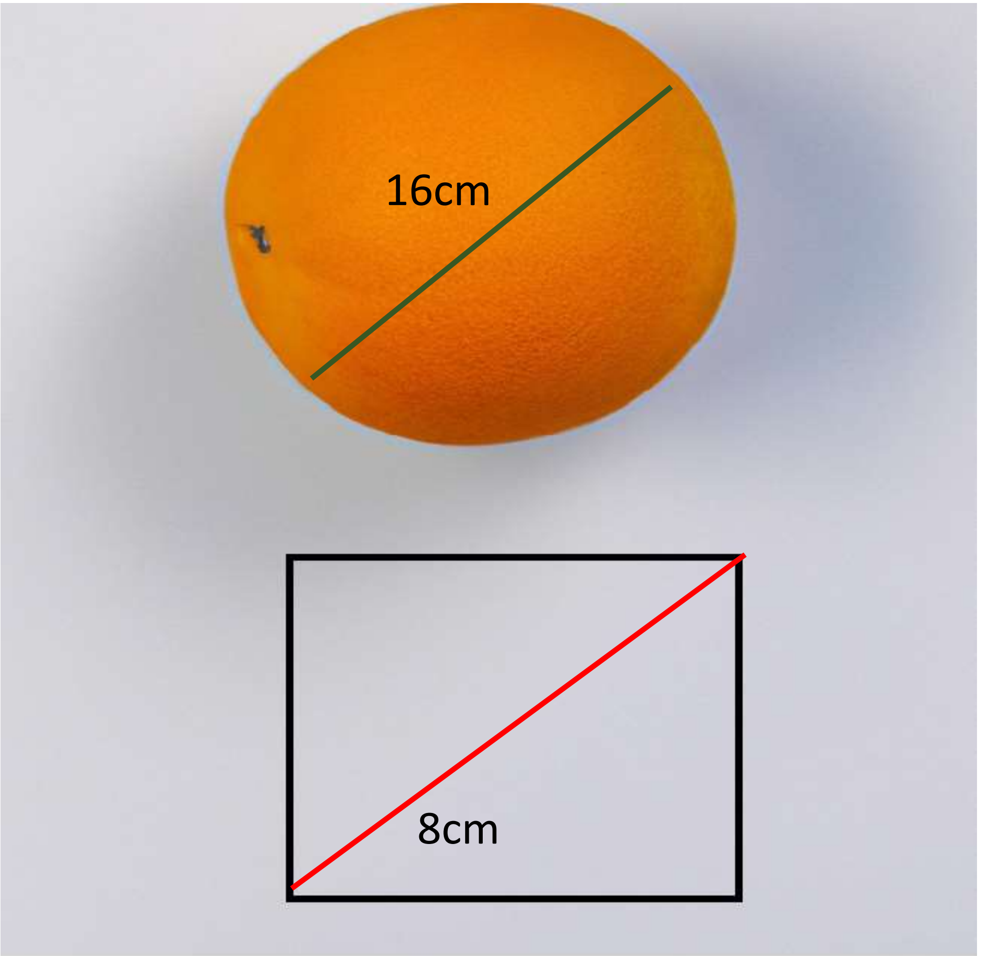


Camera calibration

Real



Camera



Orange Real = $\frac{orange\ camera * real\ square}{camer\ square}$

Defect Detection

Semantic Segmentation

Thresholding

Thresholding

Removing the background from images.

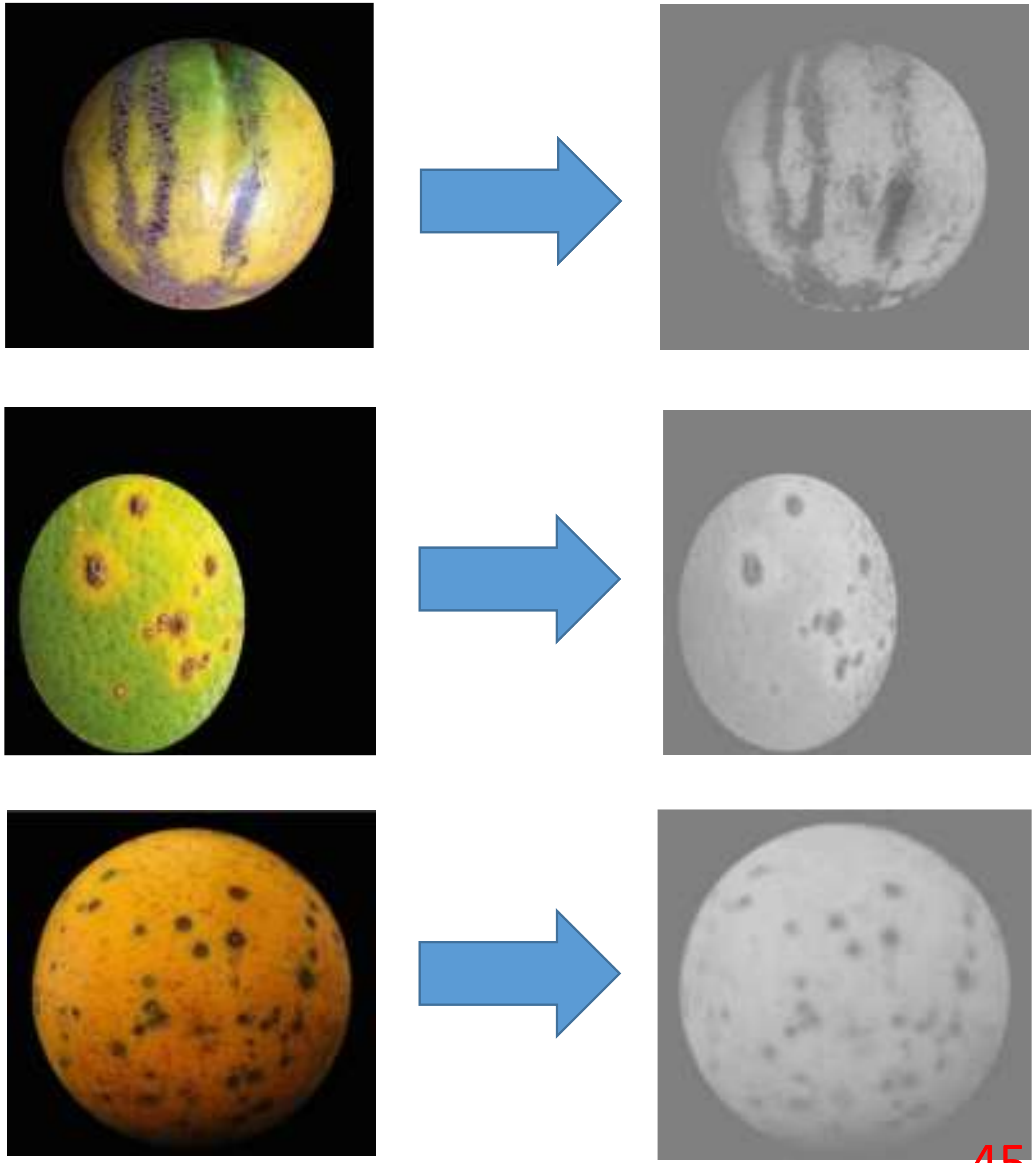
Transforming images to Lab.

Using images in b channel.

Apply Threshold in images.

First Step:

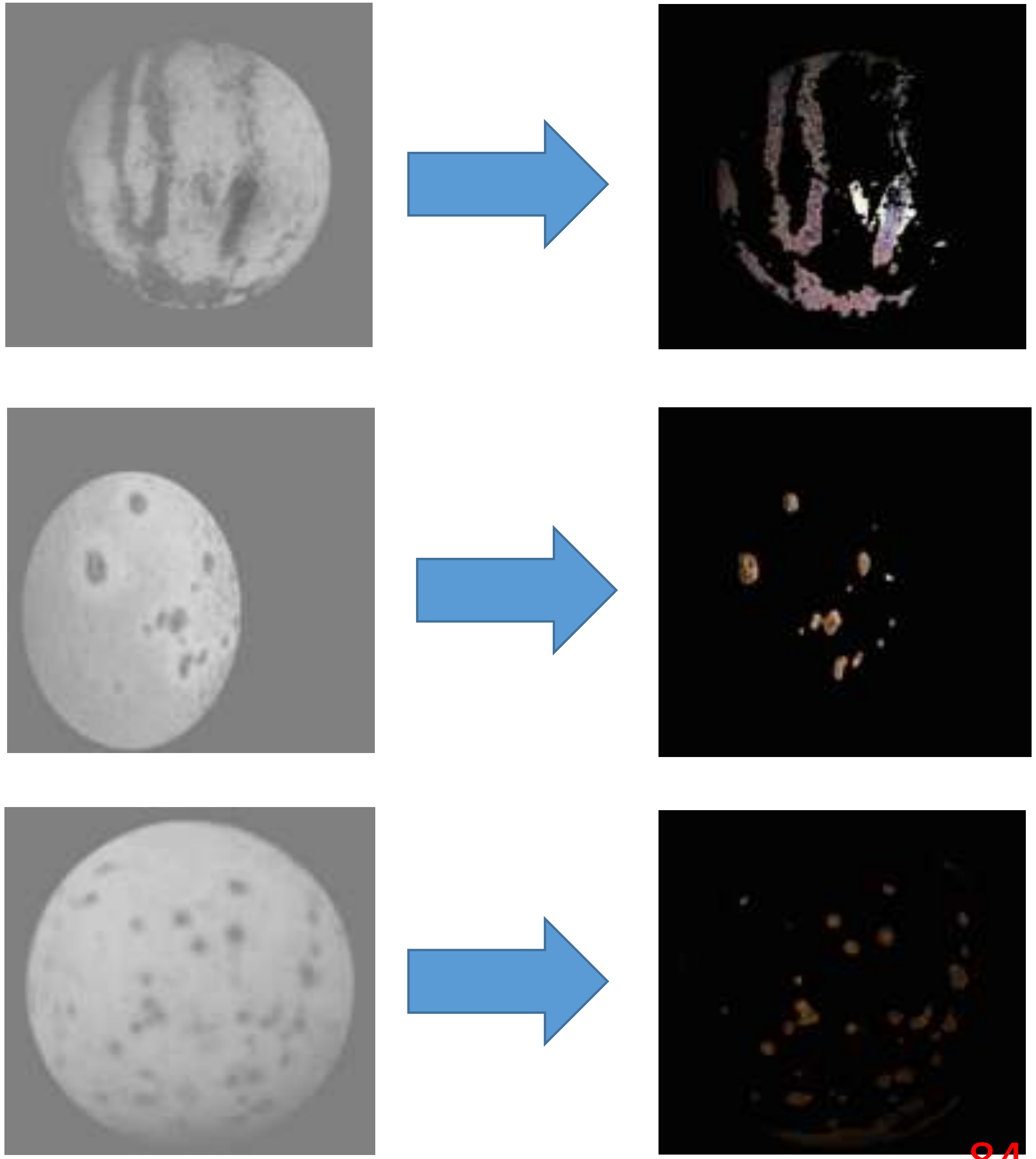
- After removing the background from images.
- We transformed images to Lab.
- Then, we used images in b channel.



Second Step:

- After using images in b channel.
- Now, we can apply Threshold:

$$\text{Threshold} = \frac{3}{4} \times median(b - channelpixels).$$



The problems of **Thresholding**

The values are not Inclusive

Stem-end is considered a blemish area

Some defects have

Semantic Segmentation

CONVOLUTIONAL NEURAL NETWORKS (CNN)

VISION TRANSFORMERS (VITS)

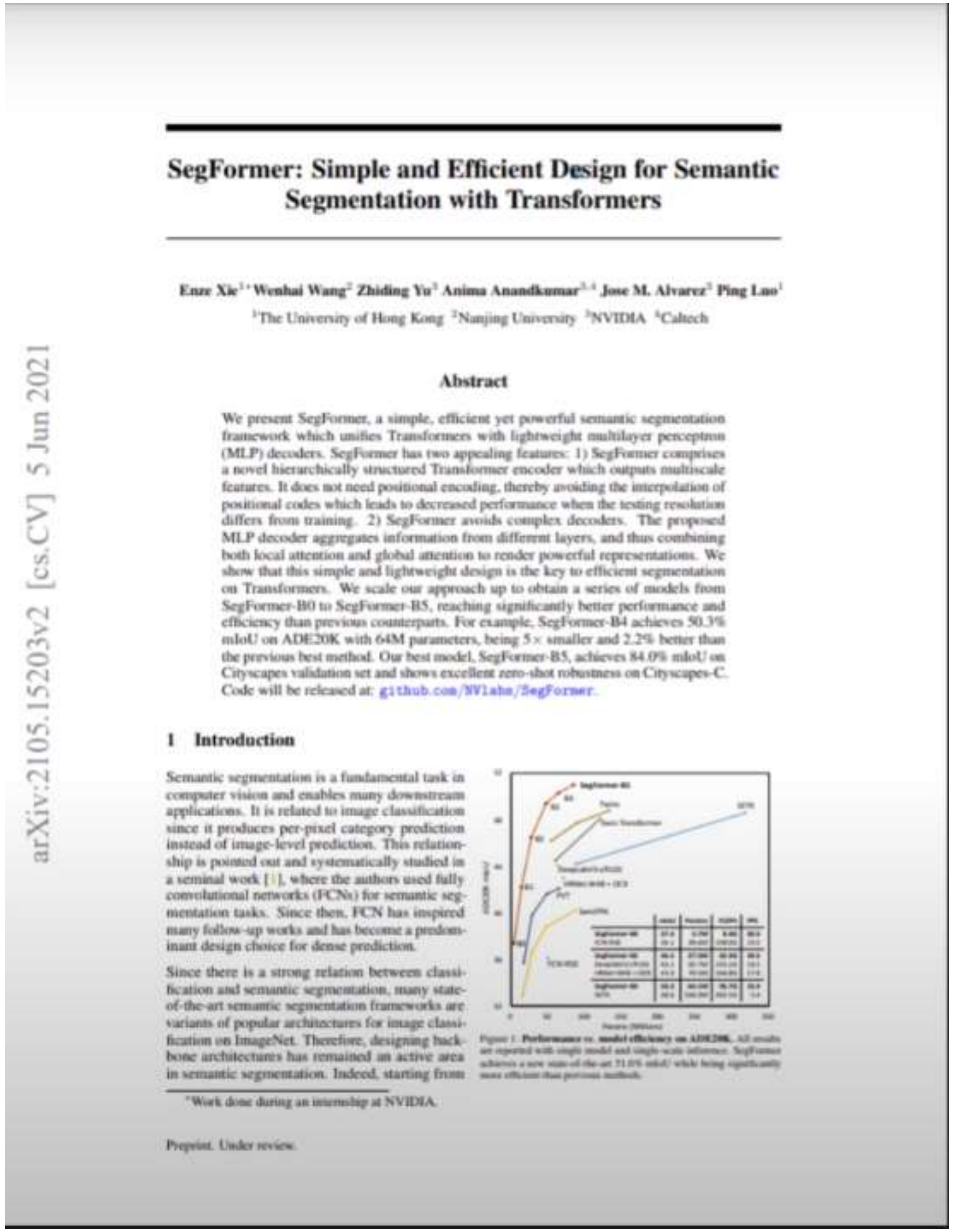
The main difference between CNNs & ViTs

Convolutional neural networks	Vision Transformers
CNNs use convolution, a “**local**” operation bounded to a small neighborhood of an image.	ViTs use self-attention, a “**global**” operation, since it draws information from the whole image.

- We used a simple, efficient yet powerful semantic segmentation framework, called **SegFormer.**

- Citation: **SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers.**

Vision Transformers



arXiv:2105.15203v2 [cs.CV] 5 Jun 2021

SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers

Enze Xie[1]* Wenhai Wang[2] Zhiding Yu[3] Anima Anandkumar[3,4] Jose M. Alvarez[3] Ping Luo[1]
[1]The University of Hong Kong [2]Nanjing University [3]NVIDIA [4]Caltech

Abstract

We present SegFormer, a simple, efficient yet powerful semantic segmentation framework which unifies Transformers with lightweight multilayer perceptron (MLP) decoders. SegFormer has two appealing features: 1) SegFormer comprises a novel hierarchically structured Transformer encoder which outputs multiscale features. It does not need positional encoding, thereby avoiding the interpolation of positional codes which leads to decreased performance when the testing resolution differs from training. 2) SegFormer avoids complex decoders. The proposed MLP decoder aggregates information from different layers, and thus combining both local attention and global attention to render powerful representations. We show that this simple and lightweight design is the key to efficient segmentation on Transformers. We scale our approach up to obtain a series of models from SegFormer-B0 to SegFormer-B5, reaching significantly better performance and efficiency than previous counterparts. For example, SegFormer-B4 achieves 50.3% mIoU on ADE20K with 64M parameters, being 5× smaller and 2.2% better than the previous best method. Our best model, SegFormer-B5, achieves 84.0% mIoU on Cityscapes validation set and shows excellent zero-shot robustness on Cityscapes-C. Code will be released at: github.com/NVlabs/SegFormer.

1 Introduction

Semantic segmentation is a fundamental task in computer vision and enables many downstream applications. It is related to image classification since it produces per-pixel category prediction instead of image-level prediction. This relationship is pointed out and systematically studied in a seminal work [1], where the authors used fully convolutional networks (FCNs) for semantic segmentation tasks. Since then, FCN has inspired many follow-up works and has become a predominant design choice for dense prediction.

Since there is a strong relation between classification and semantic segmentation, many state-of-the-art semantic segmentation frameworks are variants of popular architectures for image classification on ImageNet. Therefore, designing backbone architectures has remained an active area in semantic segmentation. Indeed, starting from

*Work done during an internship at NVIDIA.

Preprint. Under review.


road
sidewalk
building
wall
fence
pole
traffic lgt
traffic sgn
vegetation
terrain
sky
person
rider
car
truck
bus
train
motorcycle
bike
Clean
DeepLabV3+
SegFormer

Let's train SegFormer on our Custom Dataset

STEP 1

Create A Dataset

This means we need image label pairs where the label assigns a class to every pixel in the image.

Citrus melanose

Citrus black spot

Citrus canker

STEP 2

Define the Model class

A LightningModule organizes your PyTorch code into 6 sections:

```
import torch
from torch import nn
from torch.utils.data import Dataset, DataLoader
from pytorch_lightning.lite import LightningLite


class Lite(LightningLite)

    def run(self, num_epochs: int):
        model = Model(...)
        optimizer = torch.optim.SGD(model.parameters(), ...)
        model, optimizer = self.setup(model, optimizer)
        dataloader = DataLoader(MyDataset(...), ...)
        dataloader = self.setup_dataloaders(dataloader)
        for epoch in range(num_epochs):
            for batch_idx, batch in enumerate(dataloader):
              optimizer.zero_grad()
              loss = model(batch)
              self.backward(loss)  # Instead of loss.backward()
              optimizer.step()

Lite(strategy="deepspeed", devices=8, accelerator="gpu").run(10)
```

STEP 3

Train the Model

We can create the Pytorch Lightning trainer and hit the launch button!

```
GPU available: True, used: True
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
HPU available: False, using: 0 HPUs
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name  | Type                             | Params
-----------------------------------------------------------
0 | model | SegformerForSemanticSegmentation | 3.7 M
-----------------------------------------------------------
3.7 M     Trainable params
0         Non-trainable params
3.7 M     Total params
14.870    Total estimated model params size (MB)

Epoch 11: 79%  95/120 [01:03<00:16, 1.50it/s, loss=1.38, v_num=
```

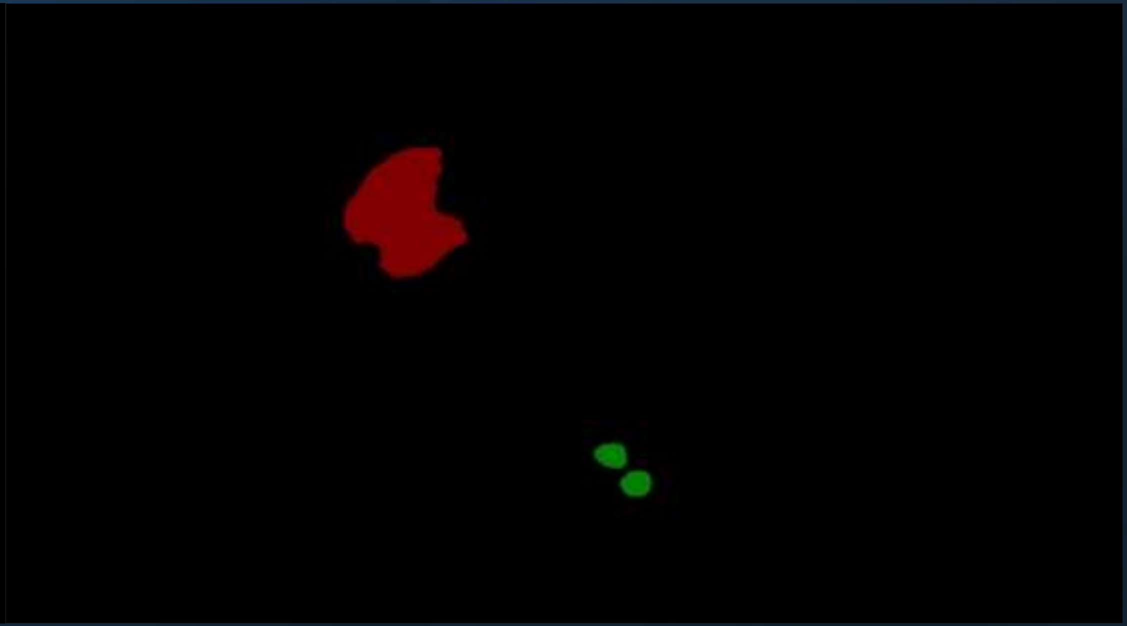
Evaluate the Model

In true Pytorch Lightning style, testing our model is a one line!

```
Testing DataLoader 0: 100%                                   1/1 [00:02<00:00, 2.24s/it]
────────────────────────────────────────────────────────────────────────────────
      Test metric             DataLoader 0
────────────────────────────────────────────────────────────────────────────────
   test_mean_accuracy      0.9405948221081628
     test_mean_iou         0.8995695843028959
────────────────────────────────────────────────────────────────────────────────
```

STEP 4

Visualize Results

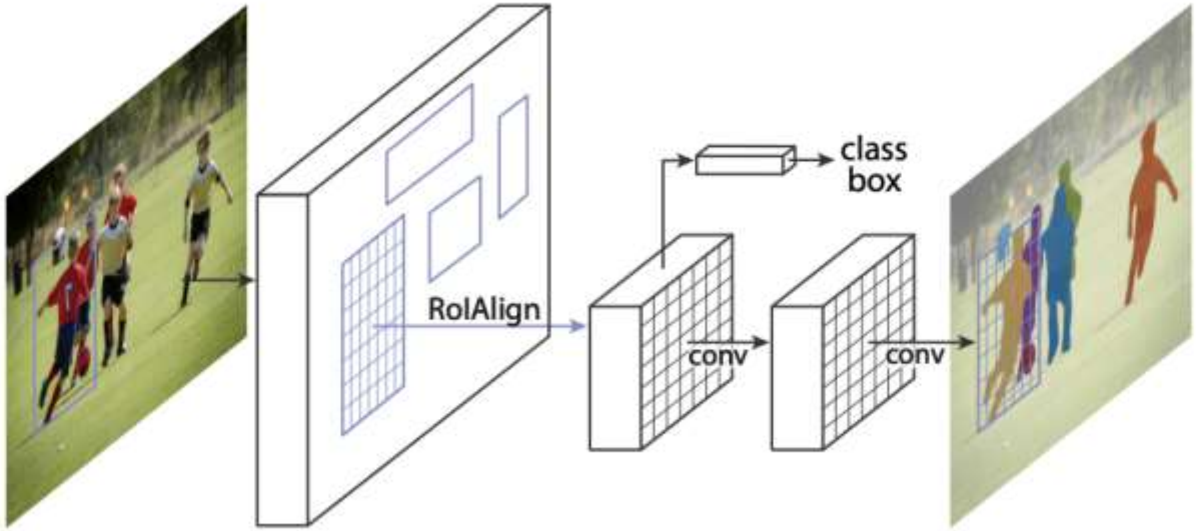


Original image (left) **SegFormer (right)**

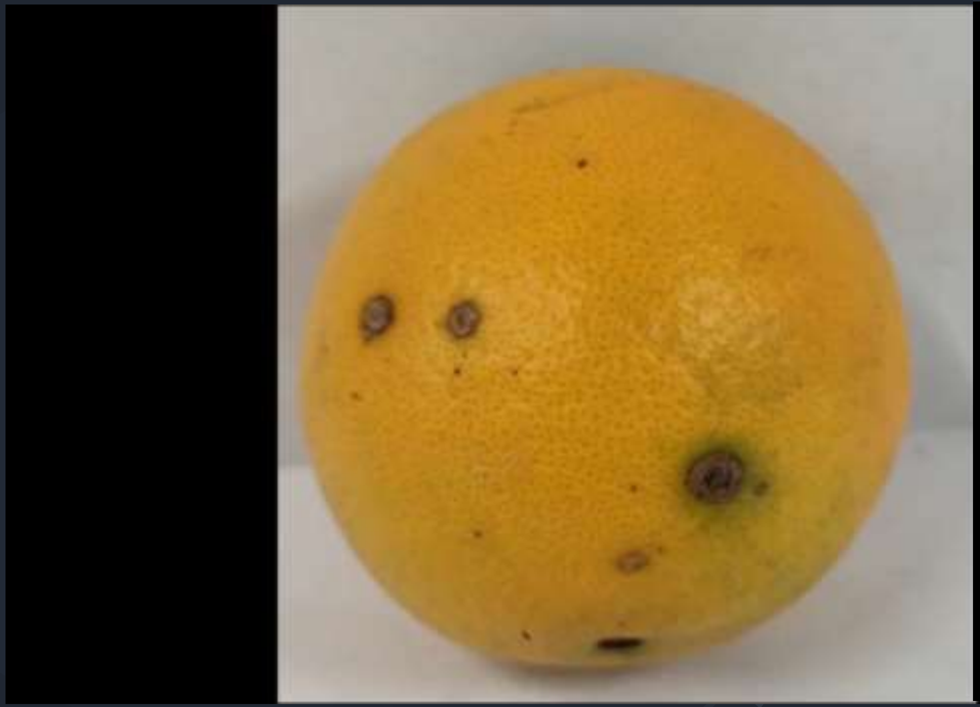
Convolutional neural networks

Mask R-CNN

- We will use Mask R-CNN.
- It's a new convolutional network proposed based on the previous fast R-CNN architecture.



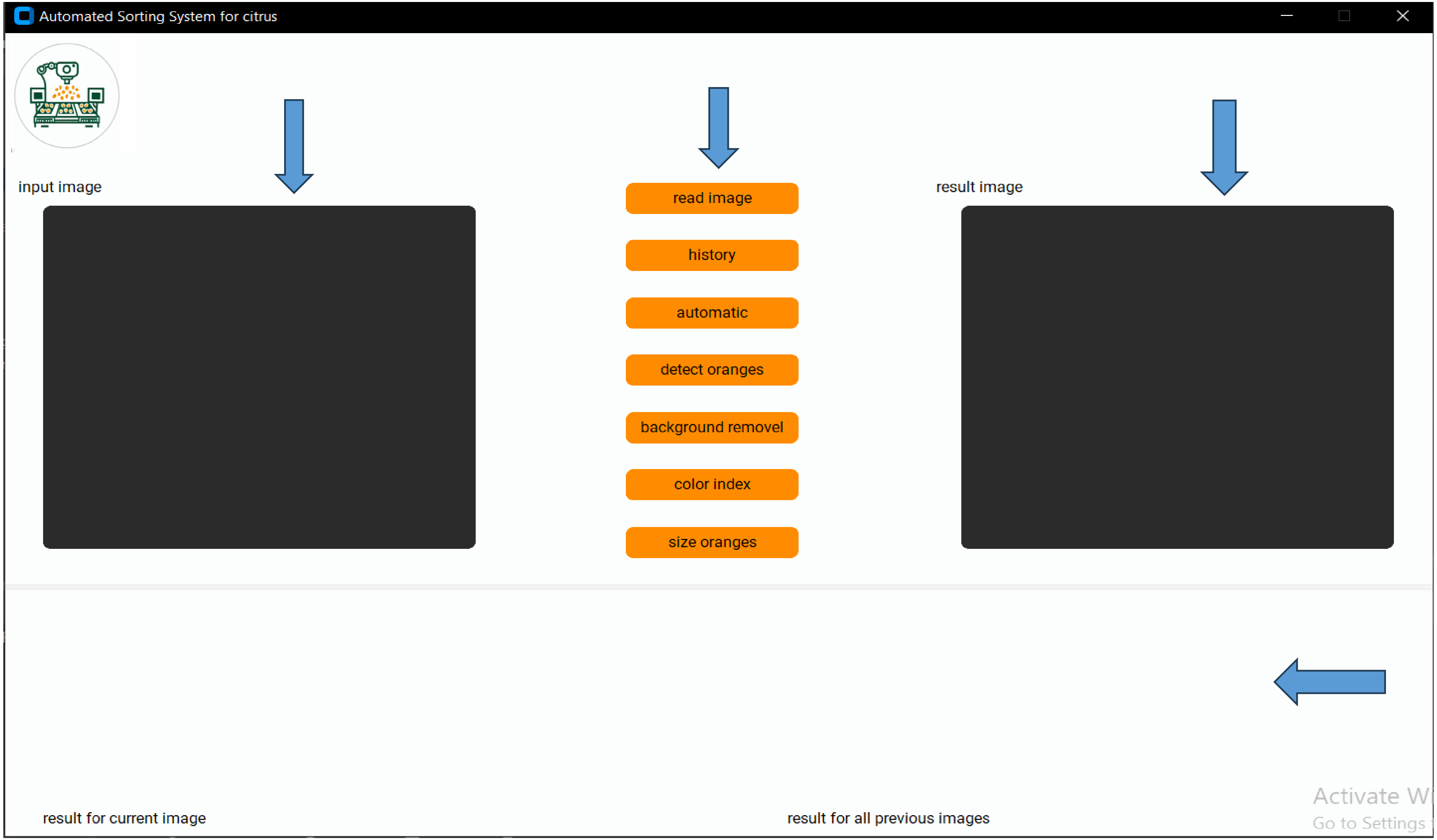
Visualize Results



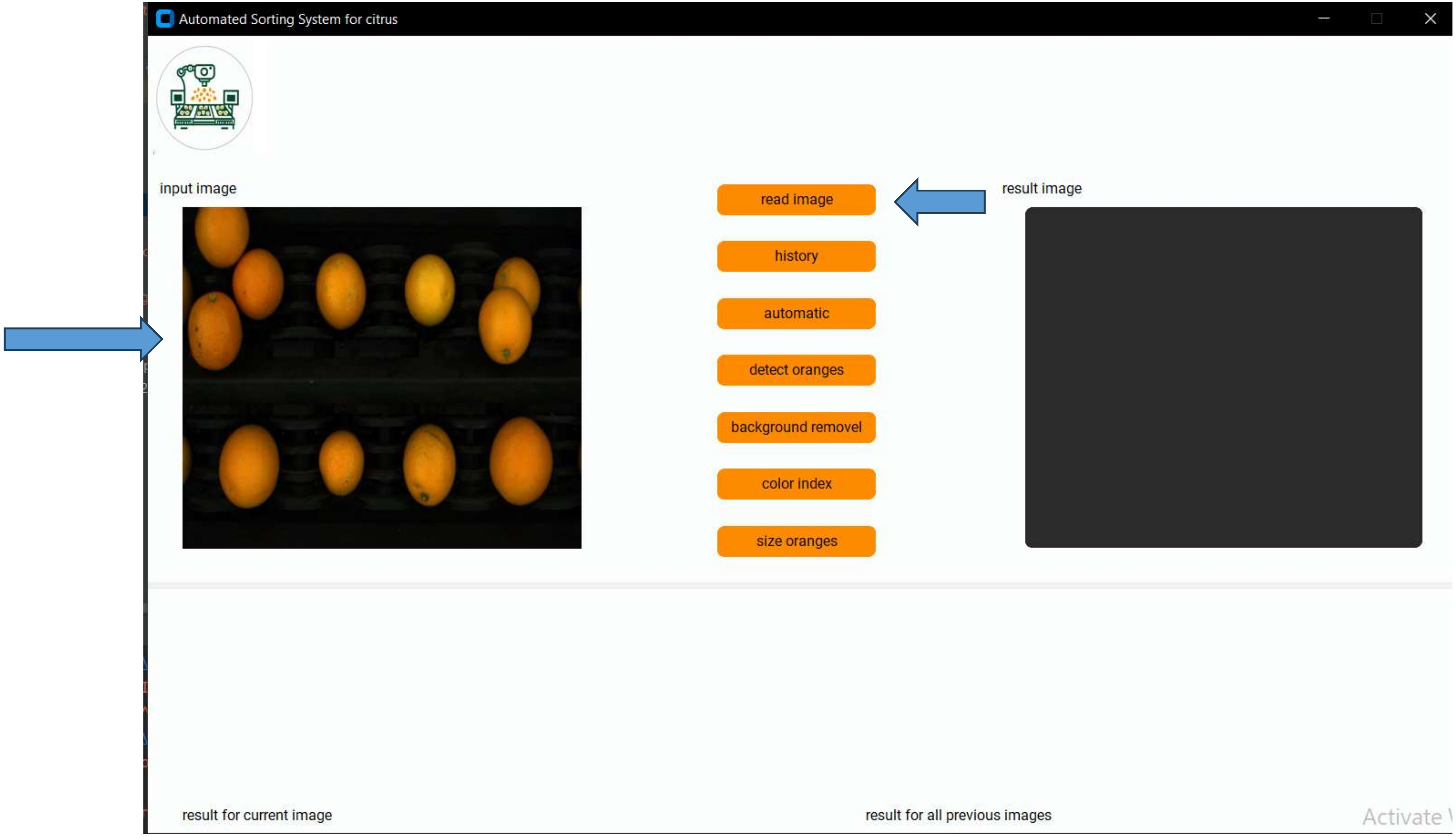
Original image

Mask R-CNN

GUI


Automated Sorting System for citrus
input image
read image
history
automatic
detect oranges
background removel
color index
size oranges
result image
result for current image
result for all previous images
Activate Wi
Go to Settings

Read image


Automated Sorting System for citrus
input image
read image
history
automatic
detect oranges
background removel
color index
size oranges
result image
result for current image
result for all previous images
Activate

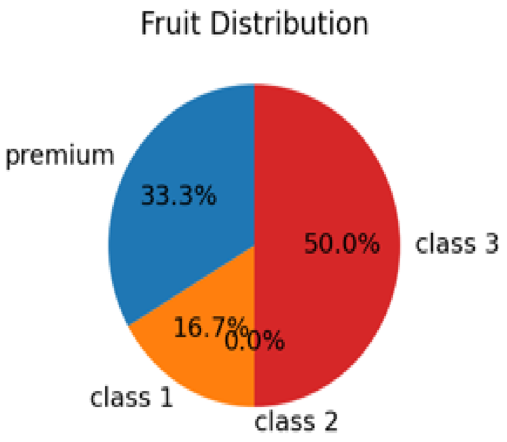
Automatic

1- the color index and size of the oranges will be displayed in two tables.

The first table shows the results for the current image

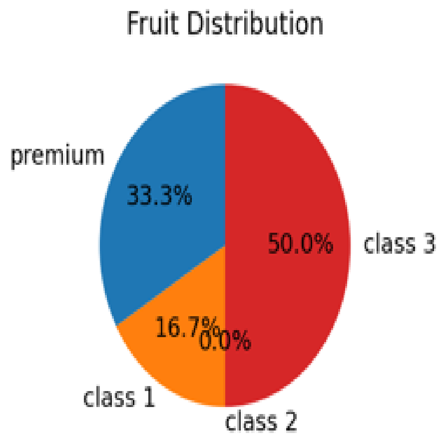
The second table shows the results for all previous images

name	class	size	area	color index	blemish area
	class3	48.1151		64.55787658311	
	class3	53.7965		62.15259819592	
	class3	53.2322		63.57140530445	
	premium	65.9162		64.38872516732	
	class1	64.8752		63.10984521742	
	premium	75.9842		62.71051975600	



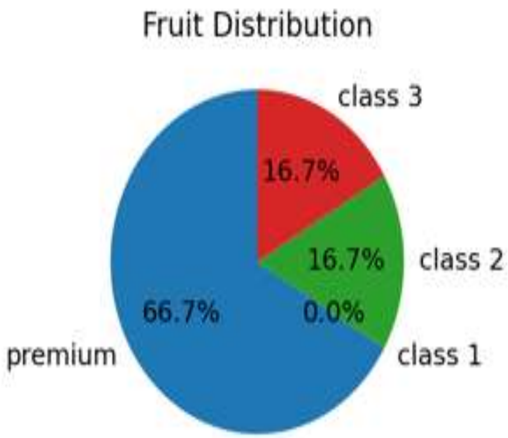
result for current image

name	class	size	area	color index	blemish area
	class3	48.1151		64.55787658311	
	class3	53.7965		62.15259819592	
	class3	53.2322		63.57140530445	
	premium	65.9162		64.38872516732	
	class1	64.8752		63.10984521742	
	premium	75.9842		62.71051975600	



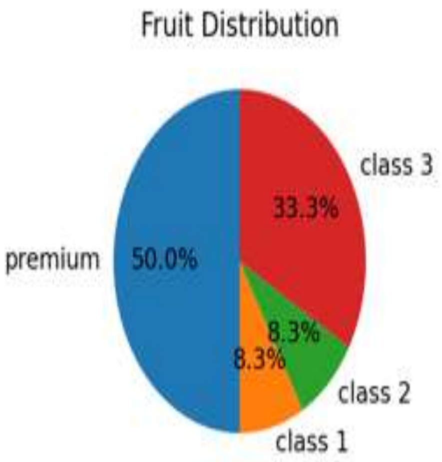
result for all previous images

name	class	size	area	color index	blemish area
	class2	57.8673		63.68714297720	
	premium	75.1706		63.19395548066	
	premium	74.3898		61.80311638649	
	class3	51.7267		64.50141171710	
	premium	72.9279		64.02219549089	
	premium	74.0228		63.12157946831	



result for current image

name	class	size	area	color index	blemish area
	class3	48.1151		64.55787658311	
	class3	53.7965		62.15259819592	
	class3	53.2322		63.57140530445	
	premium	65.9162		64.38872516732	
	class1	64.8752		63.10984521742	
	premium	75.9842		62.71051975600	
	class2	57.8673		63.68714297720	
	premium	75.1706		63.19395548066	
	premium	74.3898		61.80311638649	
	class3	51.7267		64.50141171710	



result for all previous images

Automatic

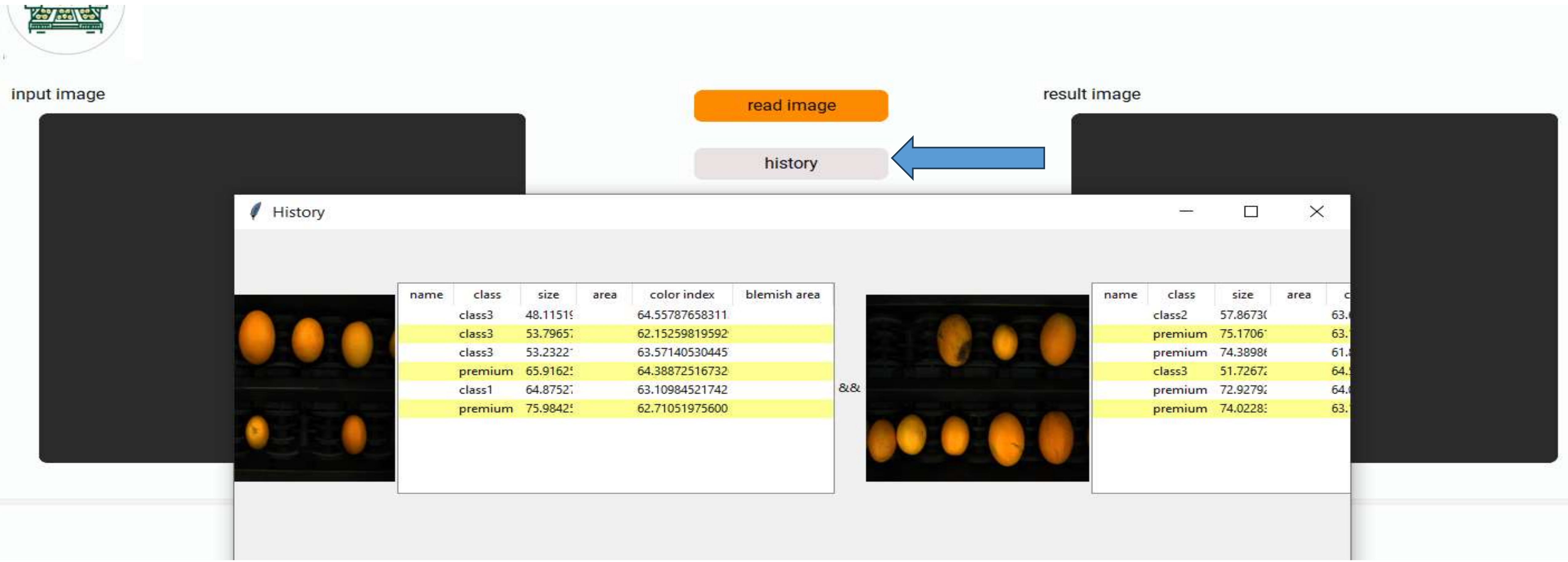
2-writing in file

1- provide a daily reports of work

2- by the type of writing

it provide a way of detect any fraud

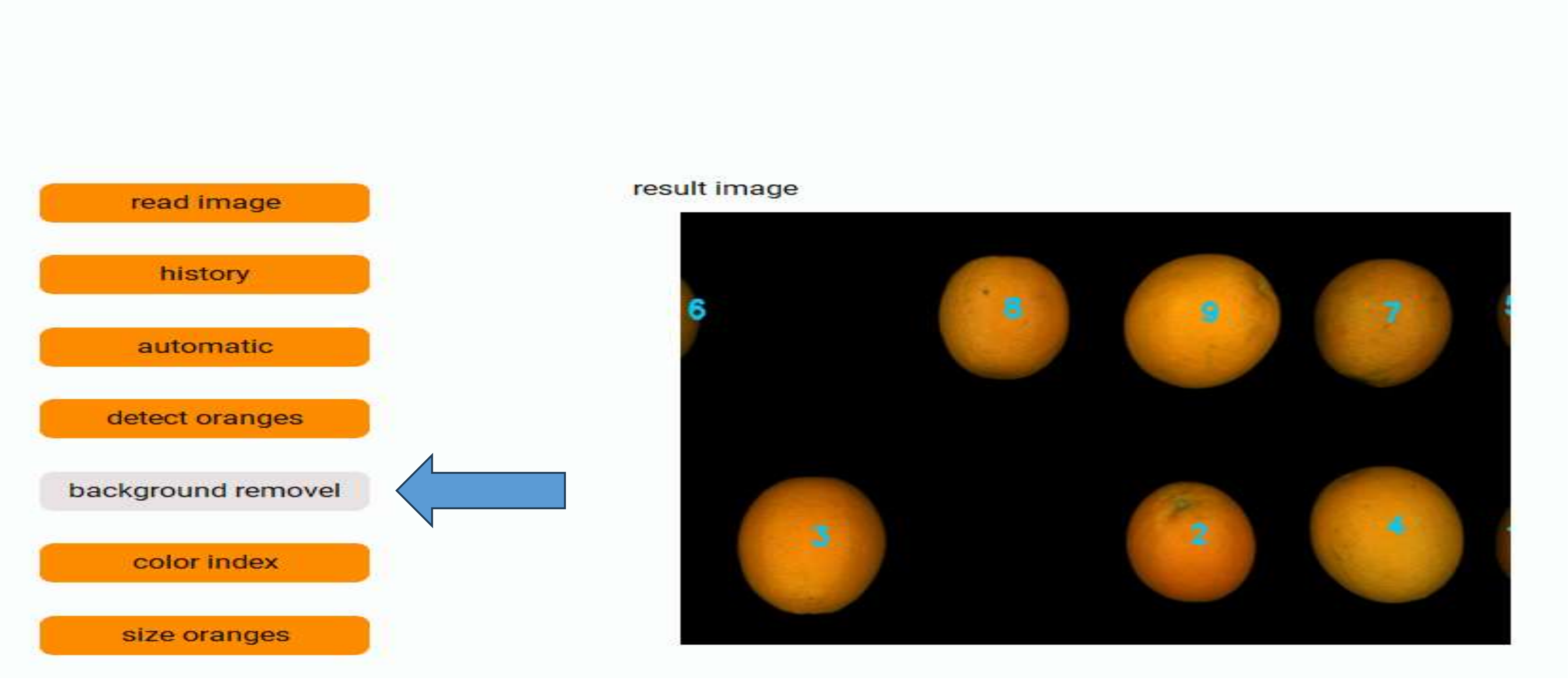
```
C:/Users/N R/Desktop/pythonProject2/1.bmp
0 class3 48.11519813537598 0 64.55787658311374
0 class3 53.796573638916016 0 62.152598195929116
0 class3 53.23221206665039 0 63.57140530445746
0 premium 65.91625595092773 0 64.38872516732671
0 class1 64.87527465820312 0 63.10984521742118
0 premium 75.98425674438477 0 62.71051975600577
C:/Users/N R/Desktop/pythonProject2/2.bmp
0 class2 57.8673095703125 0 63.68714297720027
0 premium 75.17061996459961 0 63.19395548066267
0 premium 74.3898696899414 0 61.80311638649217
0 class3 51.726722717285156 0 64.5014117171002
0 premium 72.92792510986328 0 64.02219549089963
0 premium 74.0228385925293 0 63.12157946831567
```



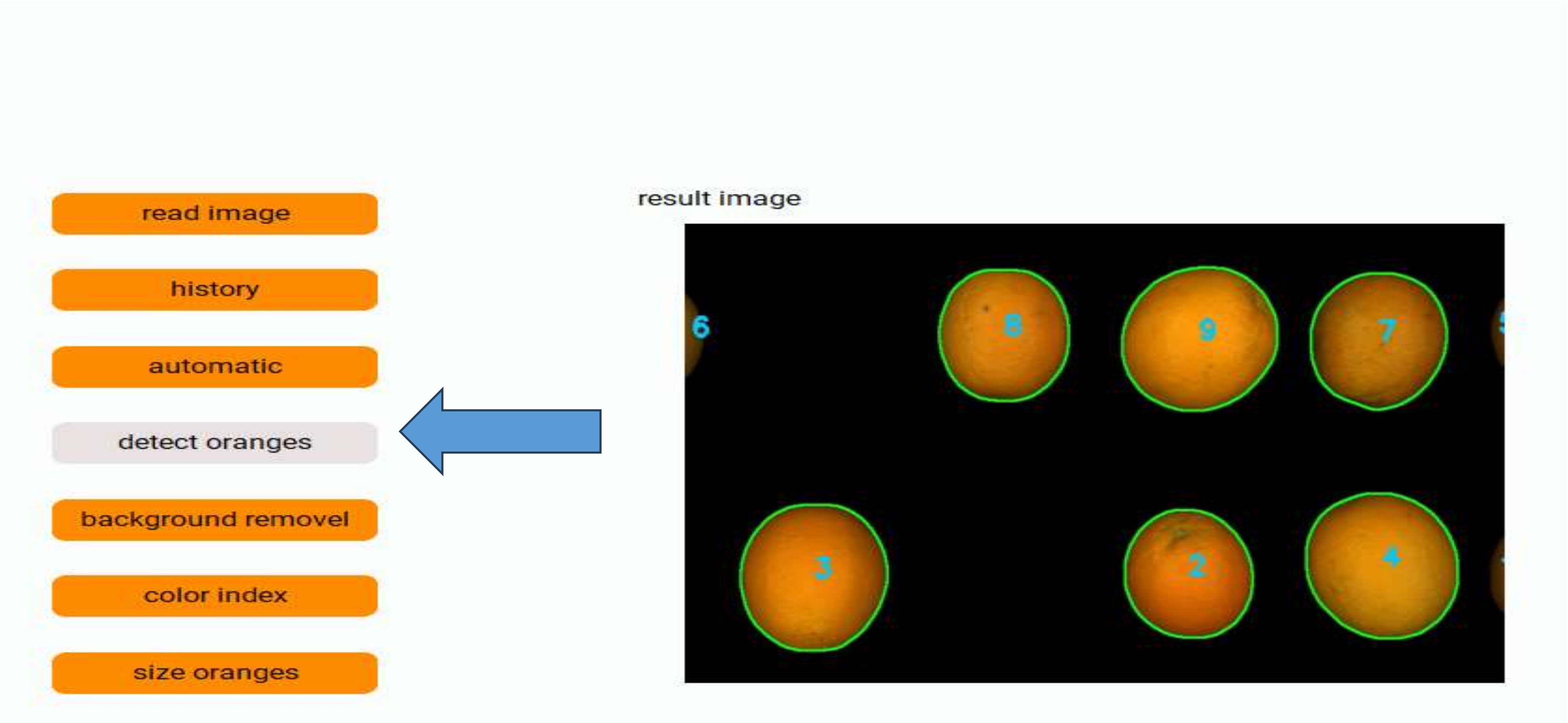
History

- The data saved in a file can be accessed by clicking the 'History' button. The history button display all saved data in the file

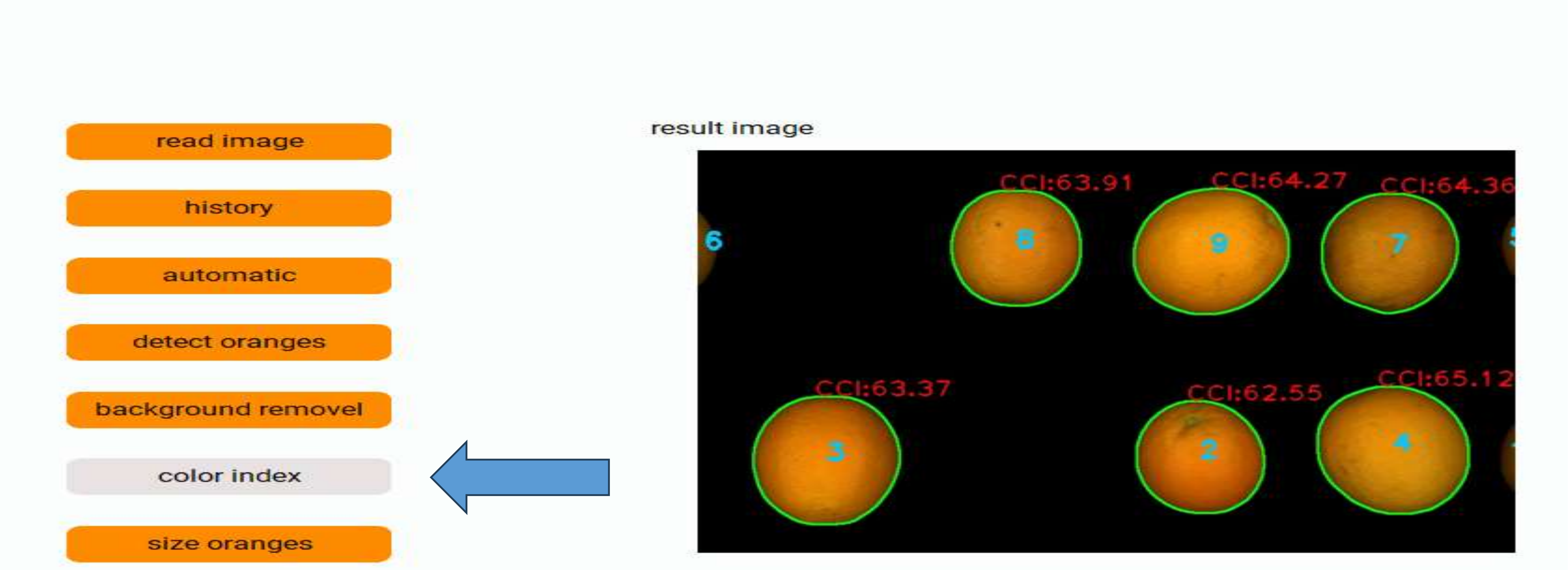
Background removal



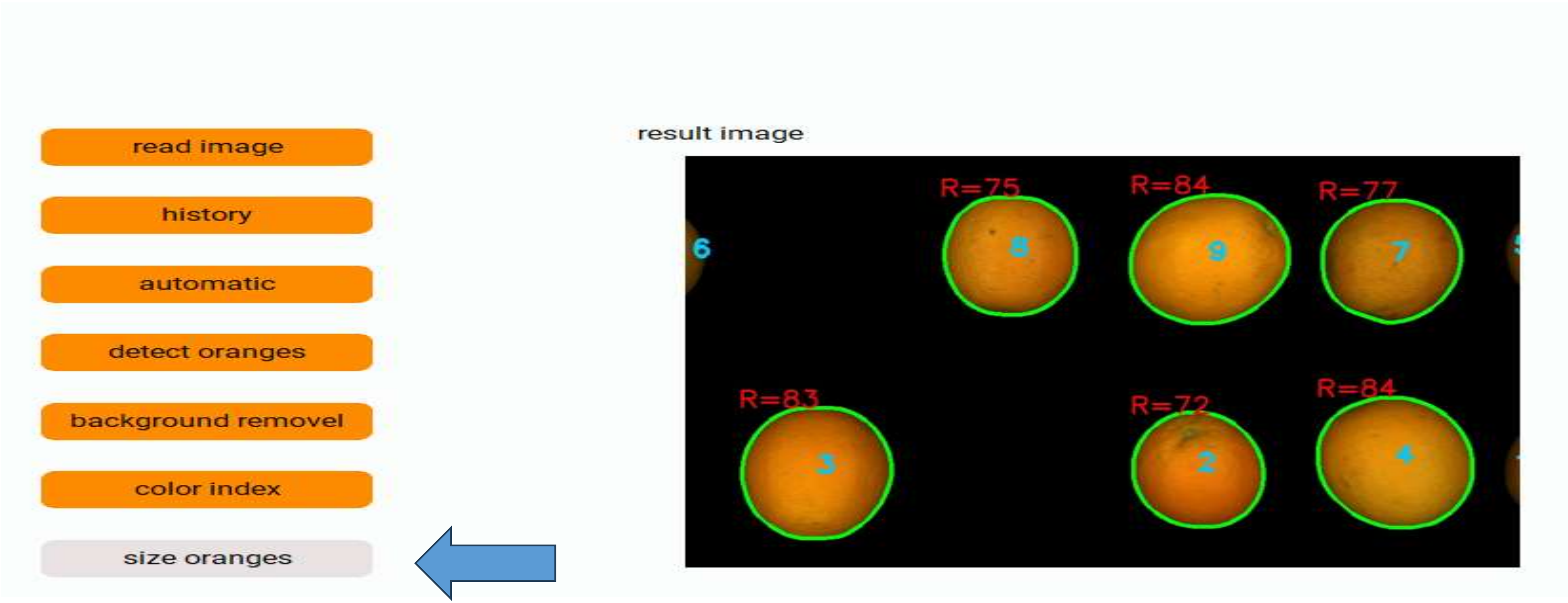
Detect oranges

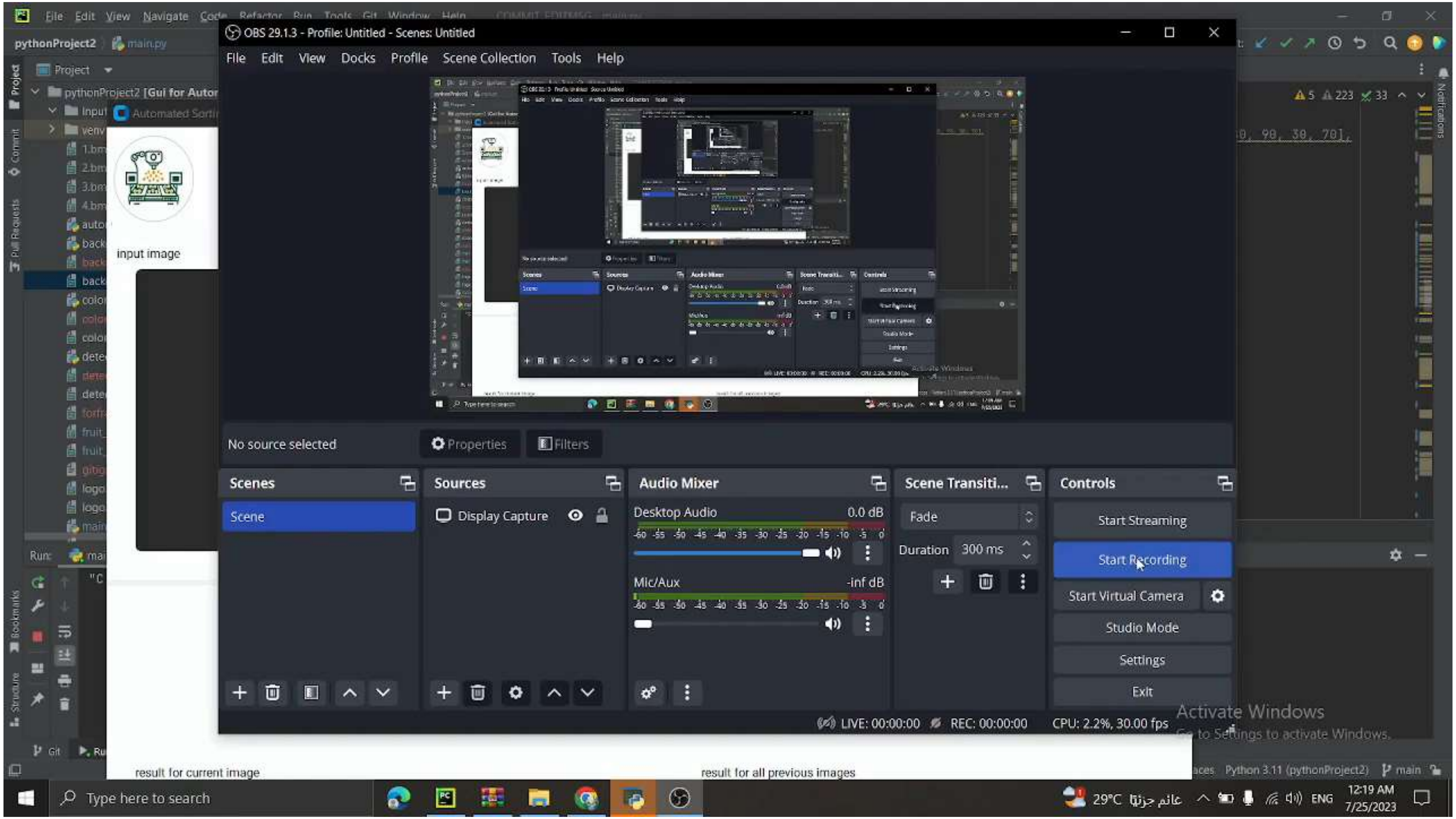


Color index


read image
history
automatic
detect oranges
background removel
color index
size oranges
result image
CCI:63.91
CCI:64.27
CCI:64.36
6
8
9
7
CCI:63.37
CCI:62.55
CCI:65.12
3
2
4

Size of oranges




OBS 29.1.3 - Profile: Untitled - Scenes: Untitled
File
Edit
View
Docks
Profile
Scene Collection
Tools
Help
pythonProject2
main.py
Project
input image
No source selected
Properties
Filters
Scenes
Scene
Sources
Display Capture
Audio Mixer
Desktop Audio
0.0 dB
Mic/Aux
-inf dB
Scene Transiti...
Fade
Duration
300 ms
Controls
Start Streaming
Start Recording
Start Virtual Camera
Studio Mode
Settings
Exit
LIVE: 00:00:00
REC: 00:00:00
CPU: 2.2%, 30.00 fps
Activate Windows
Go to Settings to activate Windows.
result for current image
result for all previous images
Python 3.11 (pythonProject2)
main
Type here to search
29°C
ENG
12:19 AM
7/25/2023

Thanks