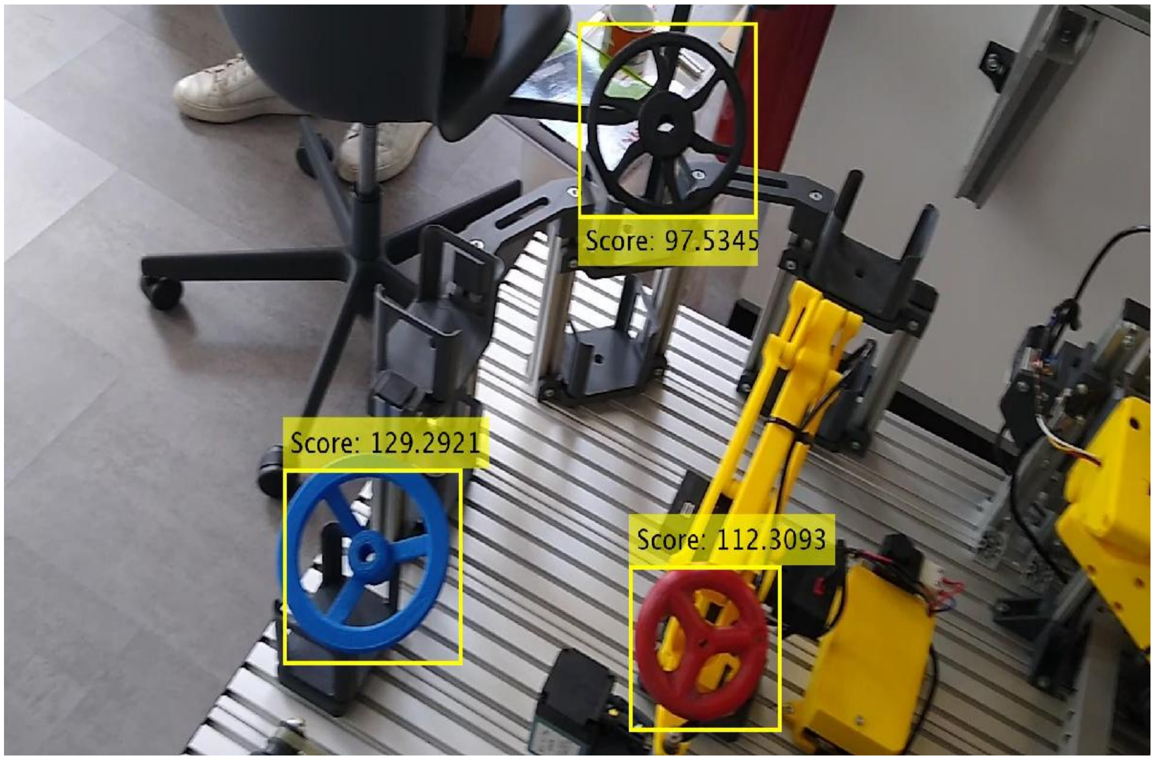
# Object Detection with Machine Learning



Our project leverages MATLAB's machine learning (ML) capabilities to automate valve detection in images. Valves play a crucial role in industries like manufacturing and utilities. Automating their detection enhances efficiency and accuracy.

Key Steps:

1. Data Acquisition: We collect a dataset of valve images, each annotated with bounding boxes indicating valve locations.

2. Data Preprocessing: Using MATLAB's Image Processing Toolbox, we resize, normalize, and augment the dataset for optimal training.
3. Model Training: Employing MATLAB's ML and Computer Vision Toolboxes, we train a model using traditional classifiers.

Outcomes:

- Accurate valve detection across diverse industrial environments.
- Increased productivity and reliability through automation.
- Reduced manual inspection efforts and associated errors, leading to cost savings and improved safety.

Future Scope:

- Fine-tuning the model for specific valve types and environments.
- Extending the system to detect other industrial components or anomalies.
- Integration with MATLAB's IoT functionalities for real-time monitoring and control.

Our project highlights the synergy between advanced ML techniques and MATLAB's toolset, driving transformative advancements in industrial automation.

## Loading the Ground Truth Data and Training the ACF Object Detector

Before detecting objects with the ACF Object Detector, we need to load the ground truth data for the valves dataset. This data is stored in a ".mat" file. First, we'll load the dataset, and then update the image file locations to match our system. To accomplish this, execute the following code section. Afterward, we'll proceed with object detection using the ACF Object Detector.

## Training the ACF Object Detector

Once the ground truth data is loaded and preprocessed, the ACF Object Detector is trained. This involves using the ground truth data to train the detector to recognize valves in images. After training, the detector will be capable of identifying valves in new, unseen images.

The function `loadAndDetectBoxes()` encapsulates these tasks and returns three important variables: `imdsTest`, `gTruthTest`, and `bboxes`. These variables contain the necessary data and information for further analysis and evaluation of the ACF Object Detector's performance.

```
load 'BoxesValves200.mat';
load 'boxestest.mat';
[imdsTest, bboxes]= loadAndDetectBoxes(gTruthTrain);
```

```
All the file paths are resolved in this groundTruth object.
detectorFile =
'acfDetector.mat'
ACF Object Detector Training
The training will take 4 stages. The model size is 76x81.
Sample positive examples(~100% Completed)
Compute approximation coefficients...Completed.
Compute aggregated channel features...Completed.
--------------------------------------------
Stage 1:
```

```
Sample negative examples(~100% Completed)
Compute aggregated channel features...Completed.
Train classifier with 598 positive examples and 2990 negative examples...Completed.
The trained classifier has 256 weak learners.
--------------------------------------------
Stage 2:
Sample negative examples(~100% Completed)
Found 75 new negative examples for training.
Compute aggregated channel features...Completed.
Train classifier with 598 positive examples and 2990 negative examples...Completed.
The trained classifier has 475 weak learners.
--------------------------------------------
Stage 3:
Sample negative examples(~100% Completed)
Found 3 new negative examples for training.
Compute aggregated channel features...Completed.
Train classifier with 598 positive examples and 2990 negative examples...Completed.
The trained classifier has 490 weak learners.
--------------------------------------------
Stage 4:
Sample negative examples(~100% Completed)
Found 1 new negative examples for training.
Compute aggregated channel features...Completed.
Train classifier with 598 positive examples and 2990 negative examples...Completed.
The trained classifier has 490 weak learners.
--------------------------------------------
ACF object detector training is completed. Elapsed time is 239.4925 seconds.
```

```

```

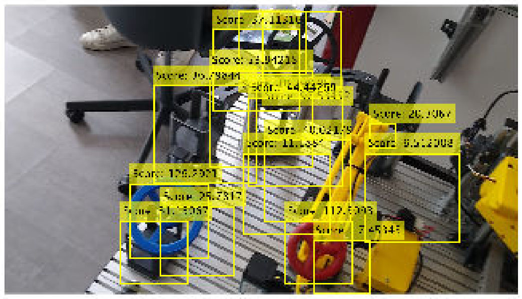
## display some image from test data

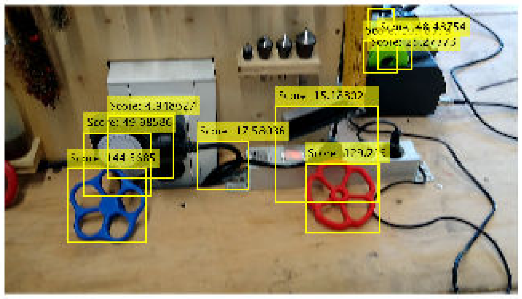
```
idx=1;
img = readimage(imdsTest, idx);
imshow(img)
```

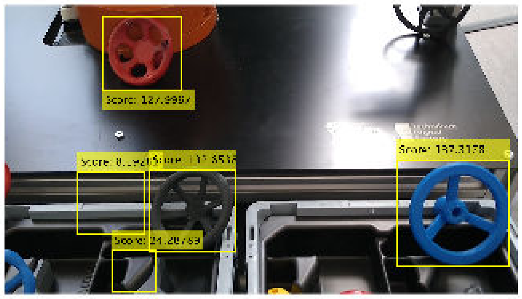
## Detect Using the Trained Detector

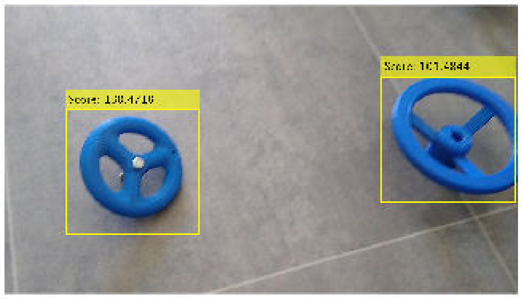
In this  phase, we embark on the pivotal task of object detection within our testing dataset. This undertaking follows an extensive period of training our ACF (Aggregate Channel Features) model, which has been meticulously tailored to recognize specific objects of interest.

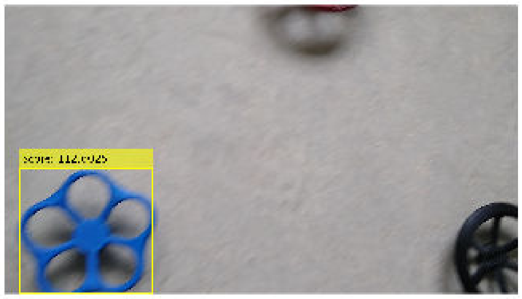
```
displayBoundingBoxes(imdsTest, gTruthTest, bboxes, 20)
```

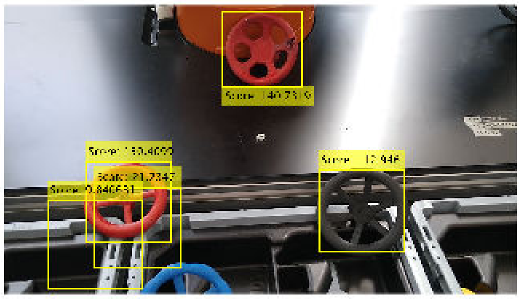

Score: 129.7907
Score: 112.5093
Score: 26.3067

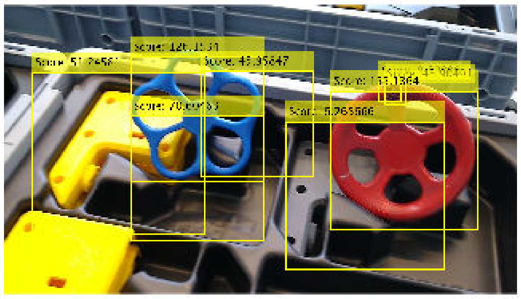

Score: 144.5585
Score: 49.98586
Score: 12.58036
Score: 15.18302
Score: 4.918527

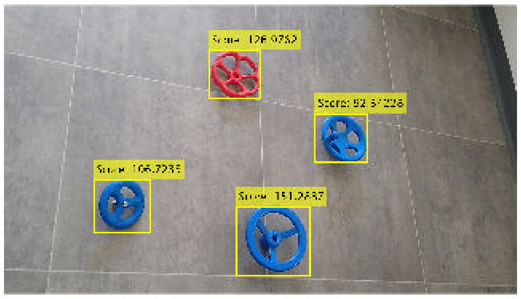

Score: 127.9987
Score: 137.3178
Score: 24.20789

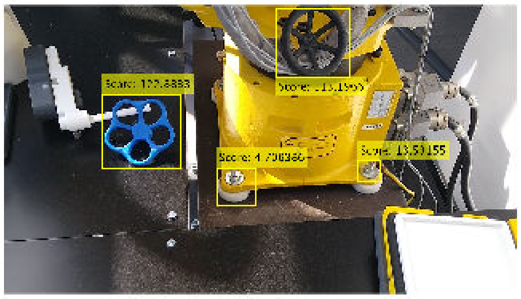

Score: 130.4710
Score: 101.4844

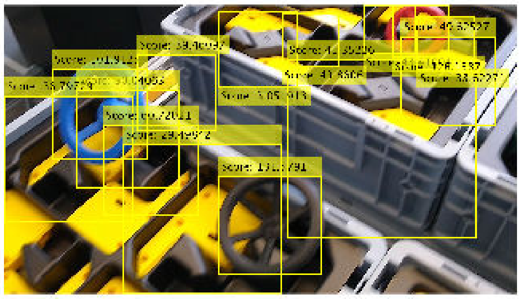

Score: 112.0925

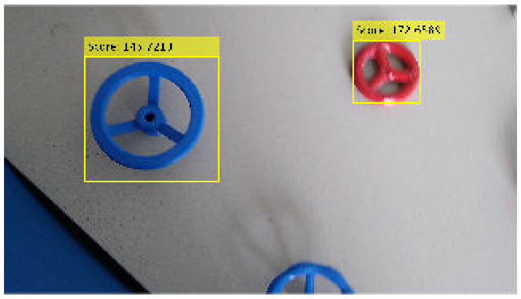

Score: 140.7319
Score: 130.4099
Score: 81.7347
Score: 0.840631
Score: 12.946

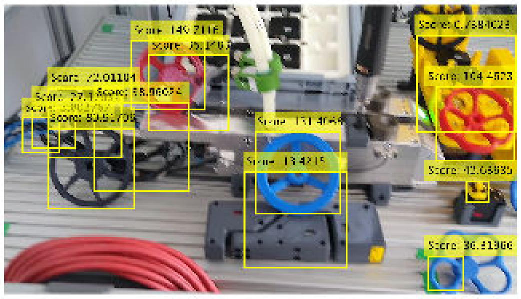

Score: 126.1531
Score: 51.24561
Score: 43.95647
Score: 153.1364
Score: 70.00763
Score: 5.765506

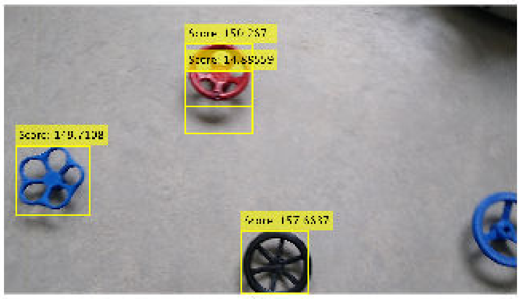

Score: 126.9752
Score: 92.57226
Score: 106.2235
Score: 151.2857

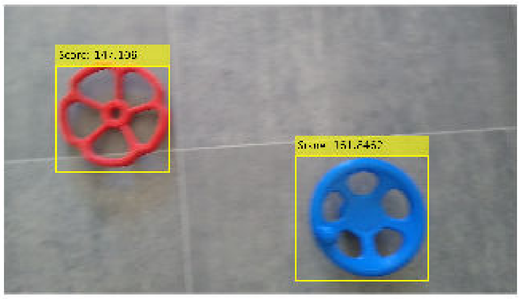

Score: 122.8883
Score: 113.1565
Score: 4.706366
Score: 13.53155

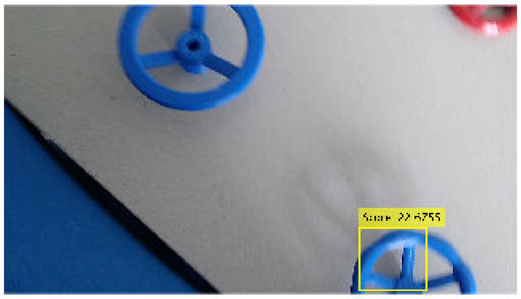

Score: 40.52527
Score: 39.46037
Score: 41.35226
Score: 101.9123
Score: 156.1387
Score: 41.8506
Score: 38.62271
Score: 38.79774
Score: 30.04053
Score: 5.057913
Score: 69.72011
Score: 29.48972
Score: 131.7791

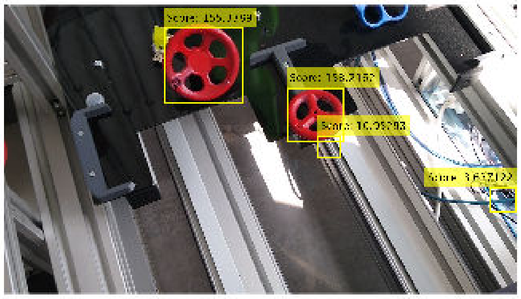

Score: 145.7210
Score: 172.6589


Score: 140.2116
Score: 0.7684028
Score: 72.01104
Score: 58.06074
Score: 80.81708
Score: 104.4573
Score: 131.4066
Score: 13.4215
Score: 42.06835
Score: 36.31966

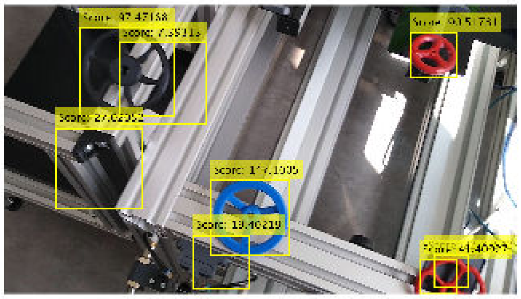

Score: 150.287
Score: 14.85559
Score: 179.7108
Score: 157.6637

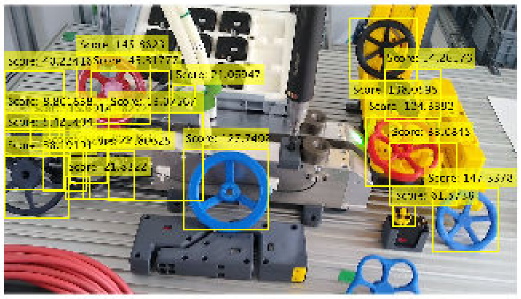

Score: 147.106
Score: 161.6462

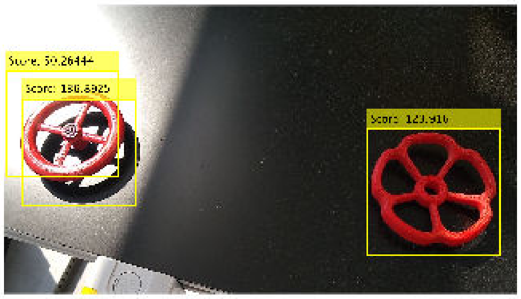

Score: 22.6755


Score: 155.7769
Score: 158.7157
Score: 16.09793
Score: 3.637122

Score: 118.8695


Score: 97.47168
Score: 7.35116
Score: 90.51781
Score: 27.02062
Score: 117.1005
Score: 19.40219
Score: 4.40097

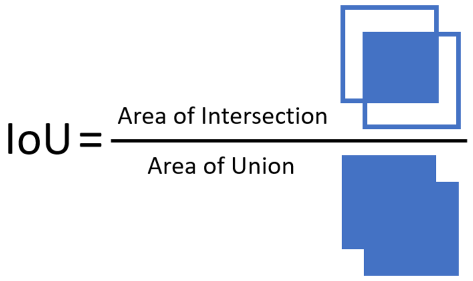



Score: 59.26444
Score: 138.6925
Score: 129.916

## omprove result Results

Intersection over Union

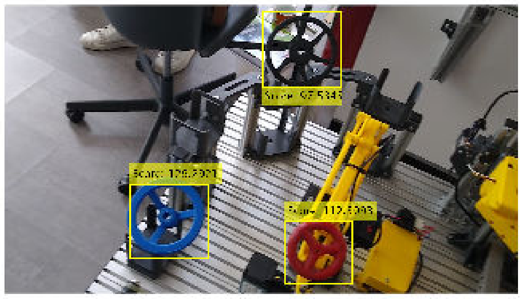
A standard method for quantifying the accuracy of a single bounding box is to calculate its overlap ratio with the ground truth, also known as the Intersection over Union (IoU).

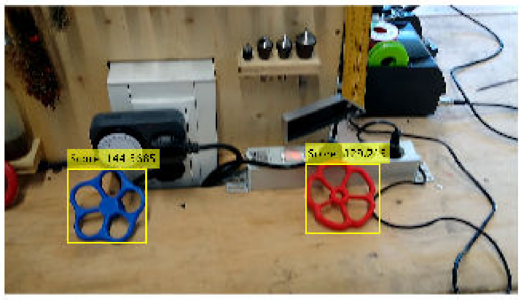


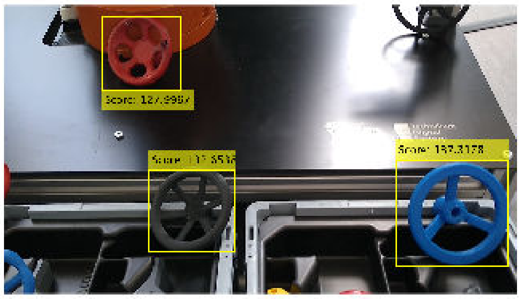
Detect only boxes with higher score values

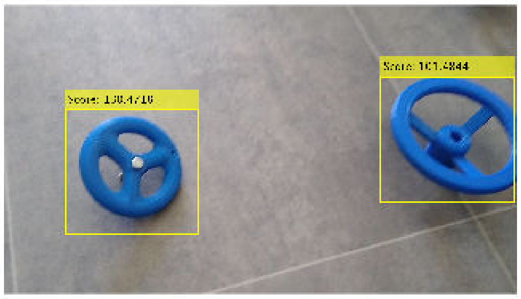
In our previous model analysis, we observed an abundance of bounding boxes in certain images, a scenario we aim to address in our current iteration. To mitigate this, we've adopted two strategic approaches. First, we employ the Intersection over Union method, enhancing the precision of our object detection. Secondly, we selectively filter out boxes with higher score values, ensuring the prioritization of more accurate detections.

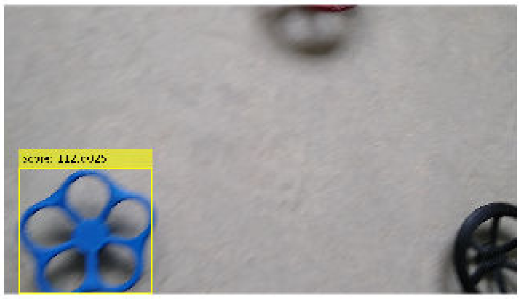
```
displayBoundingBoxesWithScoresAndIoU(imdsTest, gTruthTest, bboxes ,30)
```

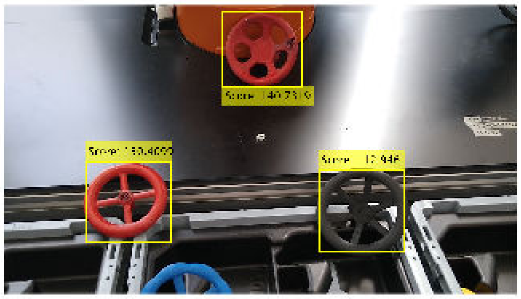


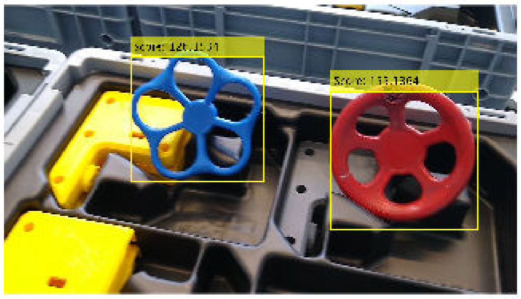


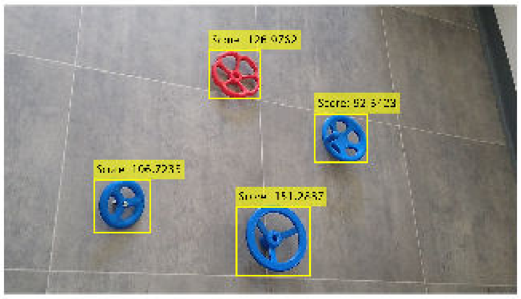

Score: 127.9987
Score: 112.8536
Score: 137.8178

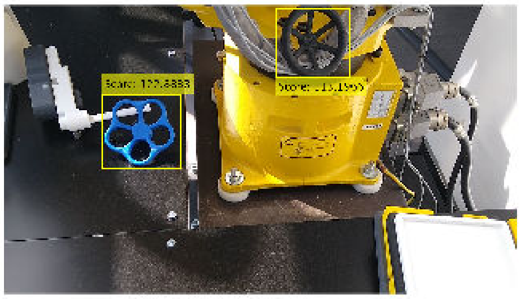

Score: 161.4844
Score: 130.4710

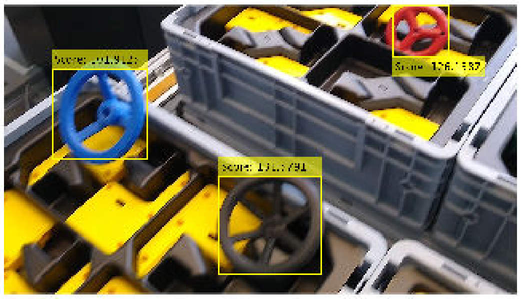

Score: 112.6925

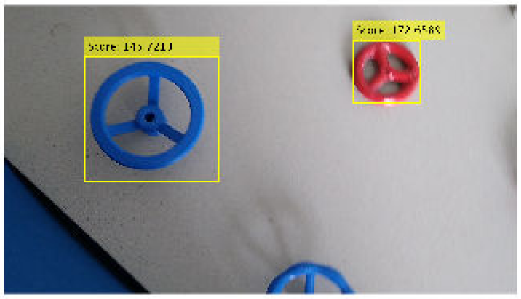

Score: 140.7319
Score: 130.4090
Score: 12.946

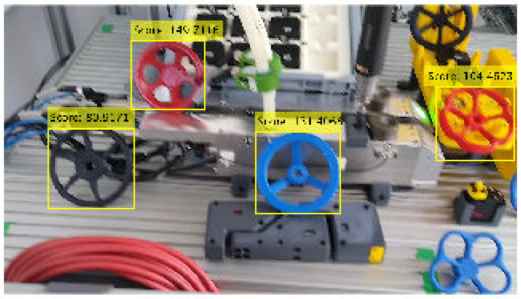

Score: 126.1551
Score: 133.1364

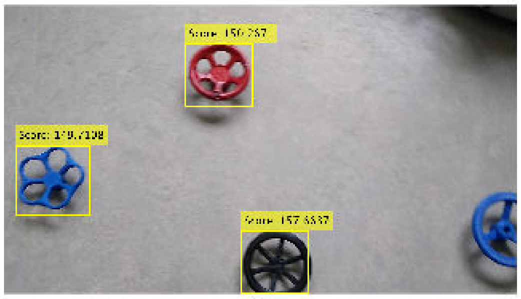

Score: 126.9752
Score: 92.5723
Score: 106.2235
Score: 151.2687

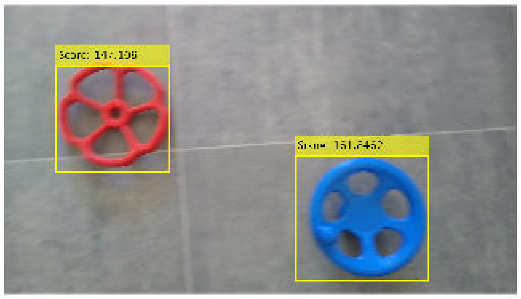

Score: 120.8888
Score: 113.1565

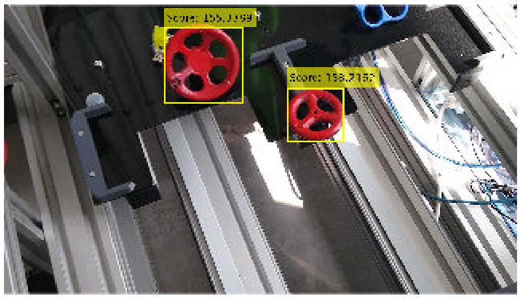

Score: 101.9126
Score: 126.1587
Score: 131.7791

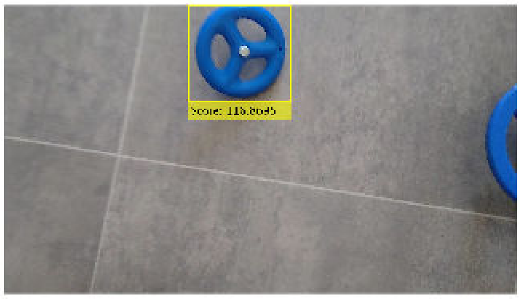

Score: 145.7210
Score: 172.6589

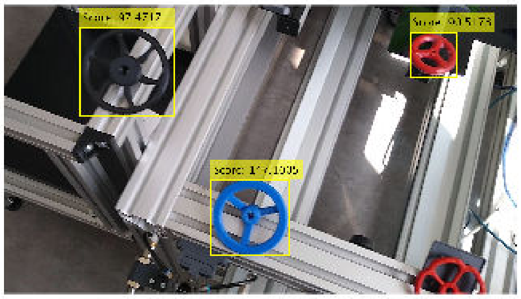

Score: 149.7116
Score: 104.4523
Score: 80.8171
Score: 131.4064

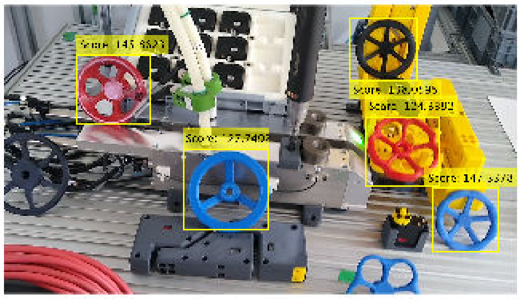

Score: 150.287
Score: 179.7108
Score: 157.6337

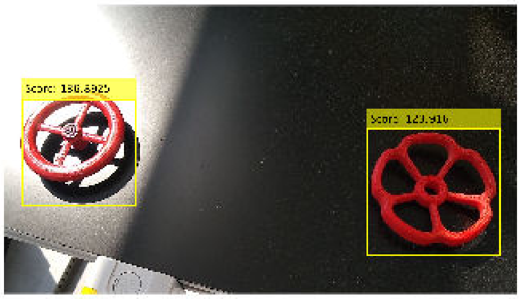

Score: 147.106
Score: 151.6452

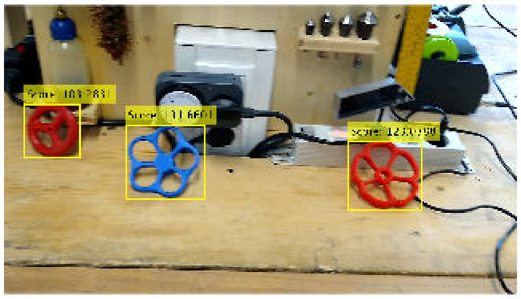

Score: 155.1759
Score: 158.7157

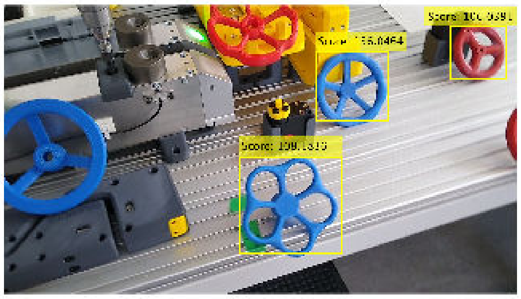

Score: 116.8695

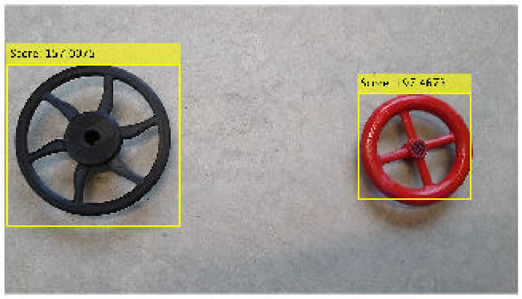

Score: 97.4717
Score: 93.5178
Score: 177.1005

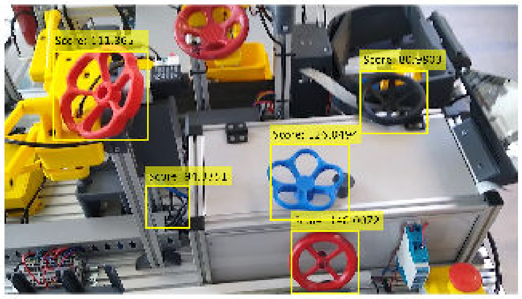

Score: 145.9823
Score: 138.9995
Score: 134.3302
Score: 127.7407
Score: 147.5378

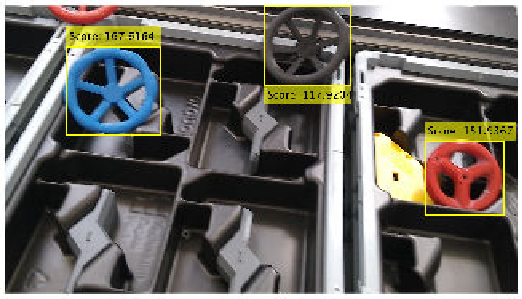

Score: 136.8925
Score: 120.916

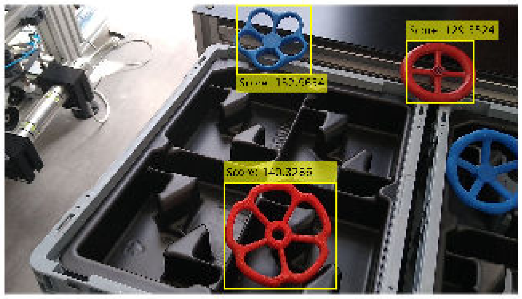

Score: 103.7831
Score: 111.6604
Score: 123.0098

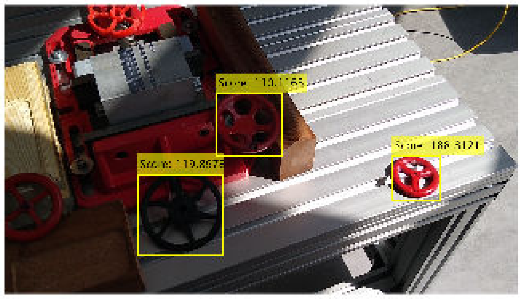

Score: 100.0391
Score: 156.0464
Score: 109.1315

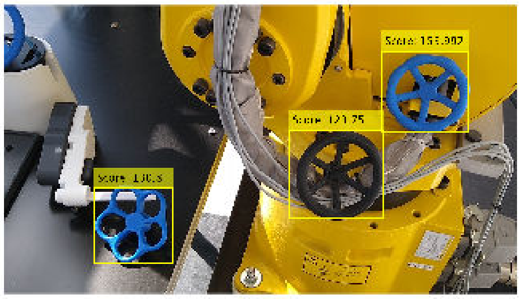

Score: 157.0075
Score: 197.4677

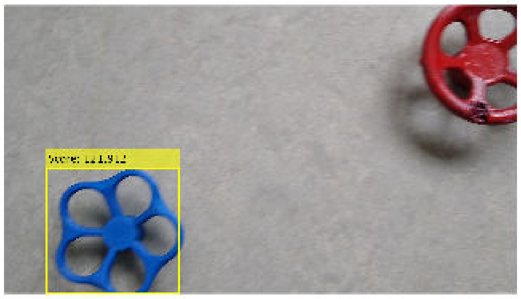

Score: 111.363
Score: 80.9800
Score: 125.0497
Score: 81.0251
Score: 140.0072

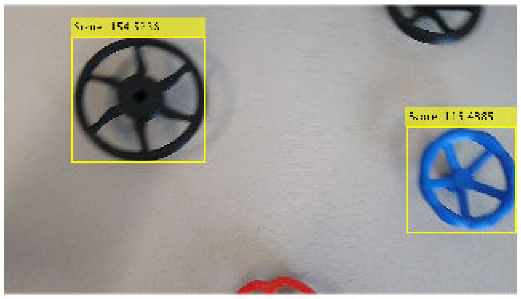

Score: 167.5164
Score: 116.9204
Score: 151.6367


Score: 129.5524
Score: 152.0554
Score: 140.3765


Score: 110.1165
Score: 188.3121
Score: 119.8675







save Result in Pdf File .

```
matlab.internal.liveeditor.openAndConvert('ObjectDetectionWithML.mlx',
'ValveDetection.pdf', 'HideCode',false);
```

## Summary and Next Steps

Summary: Despite our efforts to enhance the model, we faced inconsistencies in valve detection. Exploring a deep learning approach, especially with the YOLO algorithm, might offer better outcomes. However, deep learning training requires significant time and resources. Traditional methods, though satisfactory, may occasionally miss detections. Yet, the potential of deep learning in such projects remains promising.

Next Steps: To address these challenges, we plan to refine the model further by optimizing training data and parameters. Additionally, we aim to explore the integration of deep learning techniques, such as YOLO, to potentially enhance detection accuracy and robustness in valve detection tasks.

```

```