Milestone 13\_05\_2024

# 1.- Project Scheme:

This diagram provides a non-scaled representation of the project setup. At the centre of the label "CAMERA", the **origin of the camera frame** is located, and at the centre of the label "ROBOT", the **origin of the robot frame** is positioned. The blue circles indicate the locations where images of parts have been captured for the current dataset. The operational range of the robot, from its coordinate axis is, and .

A chart of a number of circles

Description automatically generated with medium confidence

# 2.-Project Workflow:

# A diagram of a machine Description automatically generated

Initially, we have the dataset of 24 images and the positions of the pieces located at (x, y) with x = [300, 400, 500, 600] mm and y = [-250, -150, -50, 50, 150, 250] mm.

A shadow of a person's hand

Description automatically generatedA close-up of a coin

Description automatically generatedA small coin on a white surface

Description automatically generatedA coin on a white surface

Description automatically generated

A black background with a black square

Description automatically generated with medium confidenceA white light in the dark

Description automatically generatedA white circle in the sky

Description automatically generatedA white circle in the sky

Description automatically generated

We applied dataset augmentation using "tests/dataset\_augmentation.py" from {1}.

## 2.1 Image Segmentation (U-Net Model)

A U-net model has been trained to segment masks. A downscaling factor of /4 was applied to the images, meaning the resolution captured by the camera is 1920x1080x4, but the input provided to the U-net model is 480x270x4. The model achieved an accuracy of 99.9652% and a Dice score of 98.2935% on the validation set. The prediction time for an image using an NVIDIA GeForce RTX 3060 Laptop GPU is 0.05s. The metric used for error is BCE. . And the formulas for the evaluation metrics are: **.**

The model was evaluated on a test set consisting of 255 image files of parts. This dataset includes the original images, and a dataset augmentation applying spatial transformations, horizontal flip, vertical flip, and shift, scale, rotate.

### 2.1.1 First Analysis (Segmentation Image Size)

In this initial study, we compare the accuracy, Dice score, and image segmentation time when the test images are treated in their original size and when a downscaling of four is applied. It is worth mentioning that the output image size is the same in both cases since in the case of the downscaled image, after calculating the mask of the downscaled image, we resize the image back to the original size. Although resolution is lost, in this case, what is important is to have an approximate contour of the parts, specifically the central point and an approximate point of the radius.

**Results of the first study:**

|  |  |  |
| --- | --- | --- |
| **Prediction Time** | **Accuracy** | **Dice Score** |
| INFERENCE WITHOUT DOWNSCALING (Image Height=1080px Image Width=1920px) | | |
| 3.90s | 99.9679% | 96.9028% |
| INFERENCE WITH DOWNSCALING OF 4 (Image Height=270px Image Width=480px) | | |
| 0.05s | 99.9491% | 95.2344% |

For this specific application, it is considered best to apply the downscaling. We are interested in a short prediction time without losing much precision, since we want that in case the presence sensor is activated (this is interpreted as someone approaching the table where the parts are located and therefore it is possible that the positions of the parts previously calculated have moved, the camera will take another photo and the U-net model will be rerun to calculate the position of the new parts.

### 2.1.2 Second Analysis (Model Evaluation on Imagenet-C Benchmark)

In this study, a downscaling of 4 was applied to all images, so the input for the model is an image of 270px by 480px. In this study, we want to benchmark with Imagenet C. We use the test dataset composed of 255 images to construct new Corrupted datasets applying the following functions obtained from the Imagenet-C repository (link). These functions have different levels of application intensity (from s=1, the mildest intensity, to s=5, the highest intensity). We calculate the accuracy and Dice score for each corruption at each level to see which type of conditions the model is most robust against. Based on this, we will evaluate the model. It is also worth mentioning that the functions I have used are the same as those from the U-net repository but some have been modified to accept input of an arbitrary size image and not only one of224pxx224px.  
  
We first start by evaluating the similarity of the Imagenet corruptions with the change in real conditions.

**Corrupciones:**

1. **Gaussian Noise:** Represents the effect of electronic noise on camera sensors due to poor hardware quality or low-light conditions, common in dimly lit industrial environments.

A blurry image of a person's face

Description automatically generated

1. **Shot Noise:** Simulates random fluctuations in the signal captured by image sensors, as might be caused by chemical or physical variations during the image capture process under variable lighting conditions.

A blurry image of a person's shirt

Description automatically generated

1. **Impulse Noise:** Mimics artifacts such as defective pixels or camera sensor damage, which can be caused by dust particles or dirt on the sensor or physical damage.

A close up of a person's face

Description automatically generated

1. **Glass Blur:** Podría simular la distorsión visual a través de un vidrio sucio o rayado, común en entornos industriales donde las lentes de las cámaras pueden ensuciarse o rayarse.

A blurry image of a person's hand

Description automatically generated

1. **Motion Blur:** Represents blur caused by the rapid movement of objects or the camera itself, typical in assembly lines or heavy machinery movement areas.

A blurry image of a person's face

Description automatically generated

1. **Fog:** Simulates reduced visibility due to steam or smoke, a common concern in industrial processes involving high temperatures or chemical reactions.

A grey and white photo of a television screen

Description automatically generated with medium confidence

1. **Brightness:** Simulates variations in lighting, such as sudden light changes that might occur with the turning on/off of industrial lights or movements of large objects that block light sources.

A blurry image of a person's hand

Description automatically generated

1. **Contrast:** Can represent situations where light is unevenly absorbed or reflected due to the nature of materials or atmospheric conditions, affecting the clarity of images.

A close up of a sign

Description automatically generated

1. **Elastic Transform:** Simulates deformations that could occur in the vision of cameras mounted on moving or vibrating machinery, common in environments with heavy machinery.A close up of a pipe

   Description automatically generated
2. **Speckle Noise:** Imitates noise in images obtained through coherent capture technologies such as laser or radar, which can be affected by material interference.

A blurry image of a person's face

Description automatically generated

1. **Gaussian Blur:** Represents blur caused by inaccurate camera focusing, which could result from poor focus adjustment or inadequate depth of field.

A blurry image of a person's face

Description automatically generated

1. **Spatter:** Mimics spots or splashes on the lens, as might be caused by oils, chemicals, or other fluids commonly present in industrial environments.

A blurry image of a person's face

Description automatically generated

1. **Saturate:** Represents changes in colors saturation that may occur due to extreme lighting conditions or exposures to different types of industrial lights.

A blurry image of a person's face

Description automatically generated

**Results of the second Study:**

|  |  |  |  |
| --- | --- | --- | --- |
| Corruption Type | Severity | Accuracy | Dice Score |
| **Gaussian Noise** | S1 | 99.94734954833984% | 95.20487976074219% |
| S2 | 99.94499206542969% | 95.00172424316406% |
| S3 | 99.82940673828125% | 85.97074127197266% |
| S4 | 99.25354766845703% | 58.33274841308594% |
| S5 | 97.99996948242188% | 34.27475357055664% |
| **Shot Noise** | S1 | 99.9444580078125% | 94.95559692382812% |
| S2 | 99.9212646484375% | 92.99707794189453% |
| S3 | 99.65885162353516% | 75.39686584472656% |
| S4 | 98.41059112548828% | 39.66619873046875% |
| S5 | 96.76585388183594% | 24.400529861450195% |
| **Impulse Noise** | S1 | 99.94737243652344% | 95.20829010009766% |
| S2 | 99.94140625% | 94.69189453125% |
| S3 | 99.90634155273438% | 91.77832794189453% |
| S4 | 99.5926513671875% | 71.95510864257812% |
| S5 | 98.80530548095703% | 46.611412048339844% |
| **Glass Blur** | S1 | 99.9627685546875% | 96.53606414794922% |
| S2 | 99.9674072265625% | 96.89724731445312% |
| S3 | 99.9594955444336% | 96.05304718017578% |
| S4 | 99.93972778320312% | 93.94161987304688% |
| S5 | 99.48625183105469% | 3.5452771186828613% |
| **Motion Blur** | S1 | 99.90892028808594% | 91.97746276855469% |
| S2 | 99.85832977294922% | 88.05931091308594% |
| S3 | 99.85832977294922% | 88.05931091308594% |
| S4 | 99.85832977294922% | 88.05931091308594% |
| S5 | 99.80534362792969% | 84.28385162353516% |
| **Fog** | S1 | 99.96598052978516% | 96.72893524169922% |
| S2 | 99.95116424560547% | 95.20146942138672% |
| S3 | 99.92929077148438% | 92.88938903808594% |
| S4 | 99.93480682373047% | 93.48712921142578% |
| S5 | 99.90634155273438% | 90.49625396728516% |
| **Brightness** | S1 | 99.94766998291016% | 95.23286437988281% |
| S2 | 99.94559478759766% | 95.05016326904297% |
| S3 | 99.94539642333984% | 95.03038787841797% |
| S4 | 99.9459457397461% | 95.07383728027344% |
| S5 | 99.94580841064453% | 95.05142974853516% |
| **Contrast** | S1 | 99.95494079589844% | 95.5220947265625% |
| S2 | 99.8174819946289% | 78.868896484375% |
| S3 | 99.47686767578125% | 0.0% |
| S4 | 99.47686767578125% | 0.0% |
| S5 | 99.47686767578125% | 0.0% |
| **Elastic Transform** | S1 | 99.66083526611328% | 69.26597595214844% |
| S2 | 99.43266296386719% | 48.684112548828125% |
| S3 | 99.91754913330078% | 92.52983856201172% |
| S4 | 99.91522216796875% | 92.2994613647461% |
| S5 | 99.90774536132812% | 91.64102935791016% |
| **Speckle Noise** | S1 | 99.9447250366211% | 94.9792251586914% |
| S2 | 99.94210052490234% | 94.75330352783203% |
| S3 | 99.71415710449219% | 78.53206634521484% |
| S4 | 99.29316711425781% | 59.66488265991211% |
| S5 | 98.3252944946289% | 38.43293380737305% |
| **Gaussian Blur** | S1 | 99.96513366699219% | 96.62442016601562% |
| S2 | 99.48048400878906% | 2.3572840690612793% |
| S3 | 99.476806640625% | 0.5228200554847717% |
| S4 | 99.47663879394531% | 0.04287130385637283% |
| S5 | 99.47686767578125% | 0.0% |
| **Spatter** | S1 | 99.94840240478516% | 95.29679107666016% |
| S2 | 99.9531021118164% | 95.7032470703125% |
| S3 | 99.95482635498047% | 95.85297393798828% |
| S4 | 96.89107513427734% | 25.170692443847656% |
| S5 | 87.35474395751953% | 7.6394500732421875% |
| **Saturate** | S1 | 99.96902465820312% | 97.01847076416016% |
| S2 | 99.48712158203125% | 5.245969295501709% |
| S3 | 99.92831420898438% | 93.5866928100586% |
| S4 | 93.48724365234375% | 13.8410005569458% |
| S5 | 30.639028549194336% | 1.485984444618225% |

A blue and white square with numbers on it

Description automatically generated with medium confidenceA graph of a dice

Description automatically generated with medium confidenceA blue hexagon with white text

Description automatically generatedA hexagon with blue lines

Description automatically generatedA diagram of a blue and white triangle

Description automatically generated with medium confidenceA blue hexagon with white text

Description automatically generatedA circular graph with a blue line

Description automatically generatedA blue hexagon with white text

Description automatically generatedA hexagon with blue lines

Description automatically generatedA graph of a spider plot

Description automatically generatedA graph of a pie chart

Description automatically generatedA blue hexagon with white text

Description automatically generatedA graph of a spider plot

Description automatically generated with medium confidence

**Evaluation of the results:**

The analysis of the dice scores across various corruption types and severities reveals nuanced insights into the performance of the U-NET model in simulating conditions typical of industrial environments. The dice score specifically measures the model's effectiveness in accurately segmenting the target objects, crucial for applications where precise delineation of object boundaries is required.

At lower severities, the model demonstrates commendable robustness to most types of corruptions, maintaining high dice scores. This indicates that the model can effectively handle typical variations and mild disruptions expected in many industrial settings. Particularly notable is the model's performance against corruptions like Glass Blur, Impulse Noise, Fog, Motion Blur, Brightness, Elastic Transforms, and Speckle Noise. These conditions represent various real-world challenges such as dirty or smeared lenses, transient obscuration from steam or smoke, and mechanical vibrations—all common in industrial environments. The high dice scores under these corruptions suggest that the model can reliably perform under diverse and dynamically changing conditions without significant degradation in segmentation capability.

However, the model's performance significantly dips under corruptions like Contrast, Gaussian Blur, and Saturation, especially at higher severities. This drop in dice score under these conditions is critical as it points to potential vulnerabilities in the model's application. Contrast variations can occur due to uneven lighting or reflective surfaces, common in industrial settings. Gaussian Blur simulates out-of-focus images which might happen due to poor camera calibration or sudden shifts in the object's distance from the lens. High saturation might arise in environments with vibrant lighting or when materials under inspection have inherently high colour densities.

The poor performance in these areas suggests that the model may not adequately handle scenarios where visual clarity and colors accuracy are compromised. Such deficiencies could lead to errors in segmenting and identifying crucial components in precision-driven industrial tasks, thereby affecting the overall reliability and efficiency of automated systems.

To enhance the model's robustness in these less tolerant areas, incorporating a diverse range of images that simulate these specific corruptions into the training dataset is recommended. By training the model with images featuring varying degrees of Contrast, Gaussian Blur, and Saturation, it could better learn to generalize across these challenging conditions, thus improving its segmentation accuracy and operational reliability in complex industrial environments. This approach not only broadens the model’s applicability but also fortifies it against potential failures triggered by such adverse visual conditions.

## 2.2 Post-processing

Following the generation of an image mask, we compute a set of characteristics for each masked object. These features are categorized into two groups: geometric characteristics (such as area, perimeter, centroid, radius, aspect ratio, and orientation) and morphological characteristics (including convexity and texture). **El aspecto ratio no es una Buena métrica si la pieza está en diagonal, ya que el rectángulo es paralelo a los bordes de la imagen, en este caso es mucho mejor la excentricidad.**

We begin by calculating the characteristics of the segmented pieces in the healthy dataset, determining the mean and standard deviation for each property. We then construct a function that, given a list of dictionaries (each corresponding to the characteristics of potential segmented pieces in the image), filters out the pieces that meet the standards of the healthy dataset and returns a list of dictionaries with the pieces that satisfy this threshold, specifically returning only the radius and centroid.

A white speckled background with a red circle

Description automatically generatedWhite specks on a black background

Description automatically generated

It is observed that the model performs well as it identifies the piece; however, it is excessively slow since it identifies each element in the image and filters all of them. At this point, we consider applying a technique to filter out noise before passing it to the thresholding block. Various alternatives arise at this stage:

**Morphological filtering** employs operations such as erosion and dilation to refine image segmentation by eliminating small noise points and sealing minor gaps within segmented objects. This technique adjusts the shape of structures in an image based on predefined structuring elements, which can enhance the overall integrity of the segmentation.

**The median filter** is another effective method for noise reduction, specifically targeting salt-and-pepper noise. This non-linear filter operates by replacing each pixel value with the median value from a neighbourhood defined around it, thus preserving edge features while smoothing unwanted noise. Although initially subtle, increasing the size of the structuring element and the number of iterations can intensify the filter's effect, though it may still require additional processing to achieve optimal results.

**Gaussian filtering** smooths images by applying a Gaussian kernel, which mitigates the effects of salt-and-pepper noise by averaging the pixels under a bell-shaped curve. This approach is particularly beneficial for reducing image noise while maintaining the structural integrity of the image content.

Lastly, the Watershed Transformation is a robust technique used for image segmentation, particularly effective in distinguishing between objects that are in proximity. It operates based on the topology of the image, treating pixel intensity values like a topographical map where dark values represent low elevations and bright values represent high elevations. This method identifies catchment basins and ridges in the image, thus enabling precise segmentation of intertwined or adjacent objects.

1.- **Morphological Opening and Closing Techniques in Image Processing**  
  
In our study, individual alternatives were applied and observed. Each alone proved insufficient, leading us to explore combinations of these alternatives. Among the results, two approaches stood out:

**1. Morphological Opening**

Morphological opening was performed using a 5x5 kernel likely composed of ones. This process starts with erosion, which diminishes the size of white regions by effectively removing small structures and noise less than the size of the kernel. The primary aim of erosion is to mitigate the effects of scattered noise and eliminate irrelevant small objects. Following erosion, dilation is applied, which restores larger objects to their original size while maintaining structural integrity and removing excessive details and background noise. Dilation involves applying the same kernel, which reintroduces material to the regions thinned during erosion, aiding in smoothing outlines and maintaining continuity.

**2. Morphological Closing**

Following the opening, morphological closing is applied to address internal discontinuities and small gaps within the objects. This operation also utilizes the same kernel and begins with dilation followed by erosion. The initial dilation aims to close gaps and holes, crucial for ensuring visual continuity and structural integrity of the analyzed objects. The subsequent erosion refines the shapes of the dilated objects, preserving their correct dimensions and precise contours. Morphological closing is particularly useful for resolving internal discontinuities and small voids that may not have been adequately addressed by the opening.

Subsequently, contour detection and area filtering are employed, involving the identification of object boundaries based on intensity changes. Each detected contour is evaluated based on its area, and only those surpassing a certain threshold are retained. A study at various thresholds has been conducted, and the results are displayed in the accompanying figure. Notably, the images used were corrupted with Gaussian Noise at severity level 4. The goal here is to explore the threshold at which U-Net segmentation begins to deteriorate and whether postprocessing can rectify this.

A group of images of area thresholds

Description automatically generated

Following our preliminary results using morphological operations, we observed complete segmentation of the component at a threshold of 7000 pixels, although peripherals that required filtering were still present. The execution time for the postprocessing function was recorded at 0.02 seconds.

**2. Distance Transforms and Watershed Algorithm**

To achieve clear segmentation of the component, we combined morphological opening and closing with distance transforms and the Watershed algorithm to distinguish and delineate the boundaries of objects within complex visual scenes.

Distance Transform Application After applying morphological operations and area filtering as discussed previously, a distance transform was applied to the pre-processed image using the Euclidean distance metric. This transform calculates the minimum Euclidean distance from each pixel to the nearest background, producing a gradient map that emphasizes the centrality of object masses. By setting a threshold at 70% of the maximum value observed in the distance map, a precise demarcation is established to outline areas of certain foreground. These areas are critical as they denote the central elements of the objects, effectively distinguishing them from the surrounding background.

Handling Ambiguous Regions In regions where the distinction between foreground and background is not clear, the approach subtracts the certain foreground from the dilated background to generate a binary mask identifying unknown areas. These areas are essential for guiding the Watershed algorithm as they define where the algorithm needs to compute precise boundaries between adjacent objects.

Watershed Transformation Markers for the Watershed transformation are initialized through connected component analysis performed on the 'certain foreground', with each connected component uniquely labelled. This labelling is crucial as it seeds the Watershed algorithm with initial regions from which to grow. Additionally, 'unknown' regions are set to zero in the marker array, instructing the Watershed algorithm to determine the boundaries within these zones.

The Watershed algorithm is then applied to the image converted to grayscale, refining the initial markers based on the gradient information encoded within the image structure. Throughout this process, the algorithm iteratively modifies the labels of the markers to delineate the boundaries between distinct object regions, with boundary pixels typically marked with a distinct label.

The following results are obtained:

A screenshot of a computer screen

Description automatically generated

A diagram of a graph

Description automatically generated**Results:**

A diagram of a graph

Description automatically generatedA graph of a glass blue dice

Description automatically generatedA purple hexagon with red lines

Description automatically generatedA graph of a spider plot

Description automatically generatedA purple hexagon with red lines

Description automatically generatedA graph of a pie chart

Description automatically generatedA purple hexagon with red lines

Description automatically generatedA diagram of a hexagon

Description automatically generatedA circular graph with a red line

Description automatically generatedA graph of a spider plot

Description automatically generatedA graph of a spider plot

Description automatically generated with medium confidence

## **A diagram of a machine Description automatically generated**2.1) Option A: Transformation matrices (CF🡪 RF)

CF= Camera Frame, WCF= Warped Camera Frame, RF = Robot Frame**. Red Circle**: (Control Process) Feature based Filtering, **Yellow Circle**: (Control Process) **If** the proximity sensor **not** activated 🡪 No onehas moved the pieces 🡪 Take another piece. **Else:** Take another picture and process again.

A graph with red dots and green dots

Description automatically generated

**Advantages:**

1. Fast computing time (< 1s/image)
2. Accuracy
3. Reliable process
4. Explainable
5. U-net is robust to brightness changes 🡪 Robust to industrial applications.

**Still Missing:**

1. Make new pieces (Different Shapes and sizes)
2. Improve the U-Net model for multiclass segmentation.
3. Program the control processes.
4. Improve the GitHub repository.
5. Document everything in the master’s thesis.

## 2.2) Option B: Ensemble of MLP´s

1. Standardize the data ​and rescale to the range using
2. **Train multiple MLPs** with different hyperparameters using the approach discussed in the article "**Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles**" by Balaji Lakshminarayanan et al. **Input:** . **Output:** .
3. As each model focuses on different aspects, we can calculate **the confidence as the standard deviation of the predictions from each model**. If like training set 🡪 small std 🡪 high confidence. Else: big std 🡪 low confidence.

A graph of a number of data

Description automatically generated with medium confidence

A graph of different colored lines

Description automatically generated

**Problem**🡪Solution🡪**Problem of the solution 🡪** Solution…:

1. **Low amount of data points** **so Model does not generalize well.** → Get more data points → **Low** **range of operation of the robotic arm 30cm on x, 50cm on y. Big pieces** → Make smaller pieces so we can get more data points. → **Enough data points to train an MLP (?)** → Get Data points from the transformation matrix random (x, y, z)\_CF → T\_BC → (x, y, z)\_RF (GT) and then fine-tune with Corrections (Infinite points) → **It will never be better than the accuracy of the first model.**
2. More parameters 🡪 **More Storage Issues**
3. **Higher prediction time**

# 3.- Future plans:

1. Make new pieces. (Different shape and sizes)
2. Enlarge the dataset with the new pieces.
3. U-net modification for multiclass segmentation. Edge detection with this algorithm.
4. I can try with another CV algorithm for image segmentation (SAM)
5. Benchmark the U-Net model on Imagenet-C. And the other model
6. Review the tool correction + camera deviation correction program.
7. Document the code, and re-upload it to GitHub properly documented.
8. Write the master’s thesis about the whole process for the September Call.
9. I could integrate the code written by me into the robot demonstrator of Matthias.

# 4.- Bibliography:

Master´s Thesis (Github Iñigo Aduna) : <https://github.com/adunainigo/masters_thesis.git> {1}

Robot Demonstrator: <https://github.com/MatthiasDR96/robot_demonstrator.git> {2}

Imagenet-C: <https://github.com/hendrycks/robustness> {3}

U-Net: <https://github.com/milesial/Pytorch-UNet> {4}

# 5.-Professor Corrections and Suggestions:

* Original idea of the thesis safety and robustness
* Induce corruptions with different levels of severity
* Which corruptions will have more and less impact
* Dust in the factory that is in camera
* Uncase the robot and guarantee safety
* Before the enhancement and after the enhancement
* 1.-Overall setup
* 2.- Option A and Option B (adv and disadv)
* Jupyter books

JAVIER CASTELL DIAZ GITHUB