



**Hochschule  
Bonn-Rhein-Sieg**  
University of Applied Sciences

**Master of Science in Autonomous Systems**

**– Summer Semester 2018 –**

# **Semantic Segmentation using Resource Efficient Deep Learning**

**– Report on dataset creation –**

**by**

**Naresh Kumar Gurulingan**

[naresh.gurulingan@smail.inf.h-brs.de](mailto:naresh.gurulingan@smail.inf.h-brs.de)

Matr. no. 9030384

## Contents

|   |           |
|---|-----------|
| <b>List of Tables</b>                                       | <b>4</b>  |
| <b>List of Figures</b>                                      | <b>5</b>  |
| <b>1 Overview of the dataset</b>                            | <b>1</b>  |
| <b>2 Selection of a labeling tool</b>                       | <b>2</b>  |
| <b>3 Description of the labeling process</b>                | <b>3</b>  |
| <b>4 About the artificial image generation algorithm</b>    | <b>6</b>  |
| 4.1 Motivation . . . . .                                    | 6         |
| 4.2 Working . . . . .                                       | 6         |
| 4.3 Generator options . . . . .                             | 7         |
| 4.4 Sample results . . . . .                                | 7         |
| <b>5 Splits of the dataset</b>                              | <b>8</b>  |
| <b>6 Creation of dataset variants:</b>                      | <b>8</b>  |
| 6.1 Motivation . . . . .                                    | 9         |
| 6.2 Dataset variants . . . . .                              | 11        |
| 6.3 The shades of white dataset . . . . .                   | 13        |
| <b>7 Data analysis:</b>                                     | <b>15</b> |
| <b>8 Meta-data of the dataset</b>                           | <b>15</b> |
| <b>9 Notable features of the artificial image generator</b> | <b>15</b> |
| <b>10 Conclusion and possible directions of improvement</b> | <b>15</b> |
| <b>References</b>   | <b>17</b> |

**List of Tables**

|   |   |    |
|---|---|----|
| 1 | Description of generator options . . . . .  | 8  |
| 2 | Default value of generator options and whether the options are required<br>to be set. . . . . | 9  |
| 3 | Details of the "atWork_full" variant . . . . .  | 12 |
| 4 | Details of the "atWork_size_invariant" variant . . . . .                                      | 13 |
| 5 | Details of the "atWork_similar_shapes" variant . . . . .                                      | 13 |
| 6 | Details of the "atWork_binary" variant . . . . .  | 13 |
| 7 | Meta-data of the dataset . . . . .  | 15 |

## List of Figures

|    |  |    |
|----|--|----|
| 1  | Different objects required in the dataset . . . . .  | 1  |
| 2  | Contents of the labelDefs .mat file . . . . .  | 3  |
| 3  | Tools provided by the ImageLabeler app . . . . .   | 4  |
| 4  | ROI Label Definitions window . . . . .   | 5  |
| 5  | Different folder structures . . . . .  | 5  |
| 6  | Folder structure showing different object folders in both image and label folders. . . . .   | 6  |
| 7  | Sample results produced by the artificial image generation algorithm. In each row, the image on the left shows the generated artificial image and the image on the right shows a visualization of the semantic segmentation label and object detection label. At the top of every label visualization image, the objects in the image and their corresponding colors in the visualization are indicated. . . . . | 10 |
| 8  | A similar viewpoint of bearing box ax01 and bearing box ax16 where the difference in shapes between the two objects is clearly visible. . . . .  | 11 |
| 9  | A similar viewpoint of bearing box ax01 and bearing box ax16 where they appear similar in shape. . . . .   | 11 |
| 10 | Sample results produced by the artificial image generation algorithm for the "shades of white" dataset. . . . .  | 14 |

## 1 Overview of the dataset

Since semantic segmentation using deep learning is framed as a pixelwise classification task, an image of dimensions  $H \times W \times C$  requires a ground truth of dimensions  $H \times W$ , where  $H$  and  $W$  are the height and width of the image in the dataset having  $C$  number of channels.

The scope of the dataset is to include objects associated to RoboCup @Work. The selected 18 objects are shown in 1.

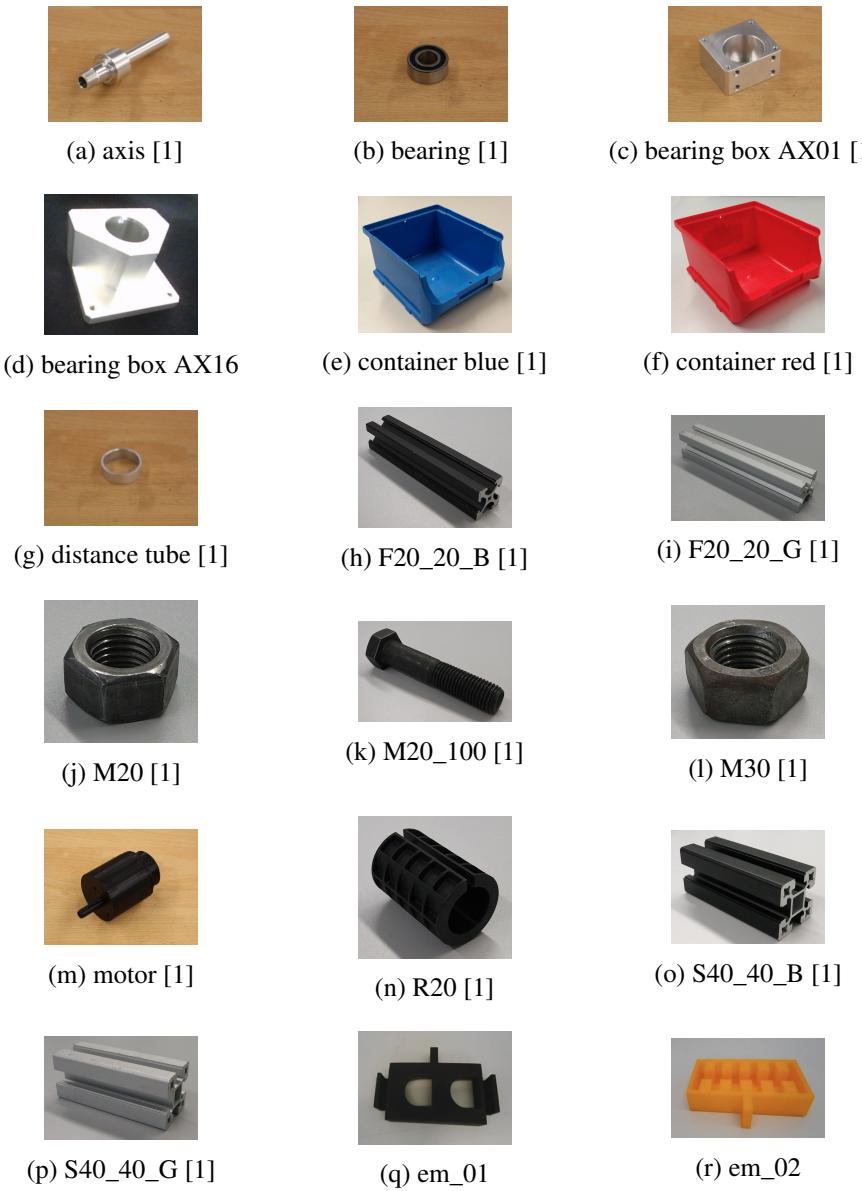


Figure 1: Different objects required in the dataset

Each of the objects were taken individually, placed on 3 different backgrounds and 30

images were taken. This lead to a total of 540 images which were to be manually labeled. Since, every pixel of the images needs to be labeled, the process of manual annotation would be time consuming. Therefore, a decision was made to first annotate the 540 images and later decide whether more images could be taken based on the effort required for annotation.

## **2 Selection of a labeling tool**

In order to reduce the time required to annotate an image, it was imperative to select a tool which is specifically designed for semantic segmentation and also provides algorithms which helps the annotator by providing labeling automation to the highest possible extent.

The following available tools were evaluated for ease of use and time taken for annotation:

- LabelMe: web based tool is public and data would also be public.
- LabelMe Matlab toolbox: yet to try..
- University bonn annotation tool:
- Pixel annotation tool (using watershed algorithm): works in windows. Seems to be useful.
- Ratsnake: tool dint seem to be useful although the website had options like superpixel suggestions.
- LabelImg: Can be used but time consuming.
- Figi: used in medical image segmentation. Has many options. Still exploring.
- Supervisely.
- MATLAB ImageLabeler available in release R2017b (Computer Vision Toolbox).

|     | 1<br>Name         | 2<br>Type | 3<br>PixelLabelID | 4<br>Description |
|-----|-------------------|-----------|-------------------|------------------|
| 1   | 'F20_20_B'        | '4'       | 1                 | "                |
| 2   | 'S40_40_B'        | '4'       | 2                 | "                |
| 3   | 'F20_20_G'        | '4'       | 3                 | "                |
| 4   | 'S40_40_G'        | '4'       | 4                 | "                |
| 5   | 'M20_100'         | '4'       | 5                 | "                |
| 6   | 'M20'             | '4'       | 6                 | "                |
| 7   | 'M30'             | '4'       | 7                 | "                |
| 8   | 'R20'             | '4'       | 8                 | "                |
| 9   | 'Bearing'         | '4'       | 10                | "                |
| 10  | 'Axis'            | '4'       | 11                | "                |
| 11  | 'DistanceTube'    | '4'       | 12                | "                |
| 12  | 'Motor'           | '4'       | 13                | "                |
| 13  | 'ContainerBlue'   | '4'       | 14                | "                |
| 14  | 'ContainerRed'    | '4'       | 15                | "                |
| 15  | 'BoundingBoxAX01' | '4'       | 9                 | "                |
| 16  | 'BoundingBoxAX16' | '4'       | 16                | "                |
| 17  | 'EM_01'           | '4'       | 17                | "                |
| 18  | 'EM_02'           | '4'       | 18                | "                |
| 19  | 'Background'      | '4'       | 19                | "                |
| ... |                   |           |                   |                  |

Figure 2: Contents of the labelDefs .mat file

### 3 Description of the labeling process

MATLAB ImageLabeler was used for the labeling process. At first, label definitions are created and exported to a .mat file. This file is used to load label definitions for all images to maintain consistency of labels. The contents of the .mat file is shown in the figure2.

The ImageLabeler app, by default, provides different tools which help create pixel-wise labels<sup>3</sup>. These tools become accessible once an image and the label definitions are loaded. A short description of the tools is given below:

- Polygon: This can be used to trace an object boundary by placing dots. Once a closed contour is created, pixels within the contour get assigned the corresponding object label.
- Smart Polygon: Can be used in a similar fashion like the Polygon tool. This tool, in addition, tries to reach out to the nearby edges of the drawn polygon.

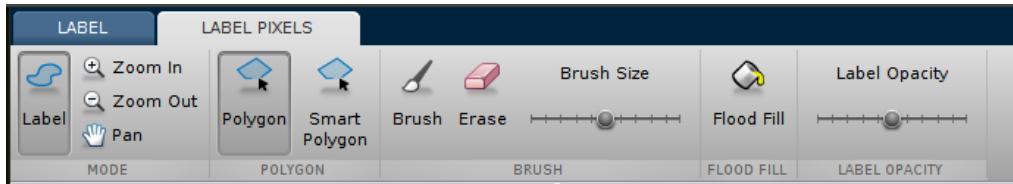


Figure 3: Tools provided by the ImageLabeler app

- Brush and Erase: Square shaped brush and eraser to either label a region or remove labels from a region. The size of the square can be changed by using the Brush Size slider.
- Flood Fill: This tool provides same labels to pixels which are similar in terms of the intensity with the selected pixel.
- Label Opacity: This tool provides a sliding bar which varies the opacity of the overlayed labels on the image. This is helpful to visualize the assigned labels.
- Zoom In, Zoom Out, Pan: These tools improve the ease of labeling by providing means to focus on particular regions by zooming and panning.

The ImageLabeler app by default assigns different colors to different objects to aid visualization. The label colors are shown in the ROI Label Definition window<sup>4</sup>.

The ImageLabeler app does not provide any tool to label all unlabeled pixels as background. In order to save time, the following workarounds have been used:

- The images taken for the dataset each have only one object in them.
- Only the object region is labeled.
- Since the ImageLabeler app does not provide any tool to label all unlabeled pixels as background, a python code which simply reads the label image and replaces unlabeled values 0 with background label value 19, was used for this purpose. The code is also used to double check the label image in order to avoid noisy labeling.

The Export Labels -> To File option can be used to save the annotations. This is done for all images individually to arrive at the folder structure shown in 5a.

The saved .mat file can be loaded into ImageLabeler again to further modify labels if required later. The 'Label\_1.png' file located in the PixelLabelData folder (as can be seen in 5a) is the label image. This image is renamed to have the same name as the image file and a folder structure as in 5b is created by using a python code.



Figure 4: ROI Label Definitions window

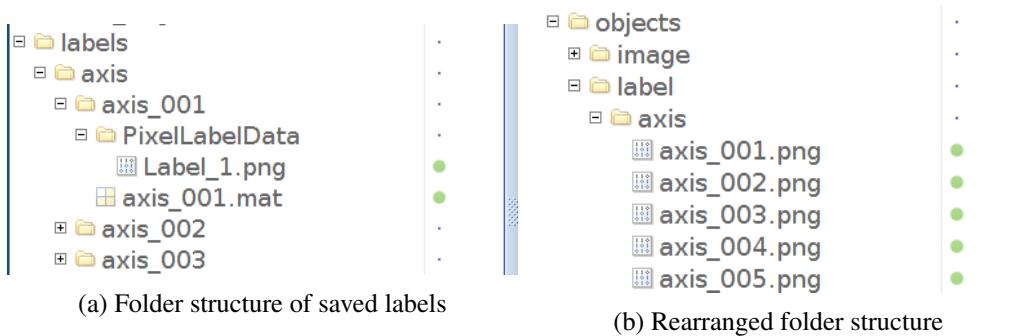


Figure 5: Different folder structures

The final folder structure is shown in 6. The image folder and label folder are similar and contain object images and corresponding label images with same names.

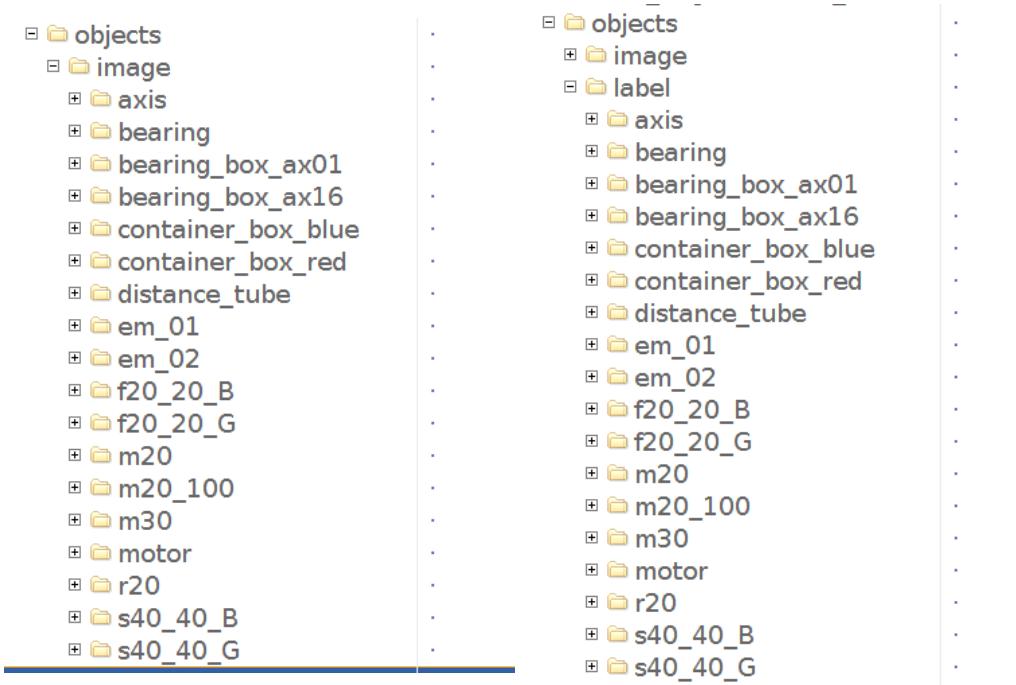


Figure 6: Folder structure showing different object folders in both image and label folders.

## 4 About the artificial image generation algorithm

### 4.1 Motivation

- Manually labeling 540 images with the described process in 3 takes roughly 2160 minutes (roughly 4 minutes per image). This is equivalent to around 4 working days. Hence, creating a large dataset with manual labeling is not feasible.
- Taking images in a variety of real world backgrounds is also time consuming.
- Labeling images with multiple objects would take an even longer time.

These drawbacks could be overcome by randomly placing objects on a variety of different background images automatically using an algorithm.

### 4.2 Working

The entire process of artificial image generation can be divided into 6 broad steps:

- **External interface:** An interface to obtain possible parameters to control the generation process.
- **Get backgrounds and data:** Fetch all the backgrounds, images and corresponding labels from the provided respective background, image and label paths.
- **Get object details:** Fetch details regarding every object and its different scales. The details include information regarding object locations in an image, object values, label values, the object name, points in pixel space denoting a bounding rectangle around the object, and object area.
- **Generate augmenter list:** Every element in the augmenter list denotes an artificial image and contains information including the chosen background image, the number of objects to place in the artificial image, which objects from the object details list are selected and locations in pixel space where the selected objects need to be placed.
- **Generate artificial images:** Based on every element in the augmenter list, artificial images are generated and corresponding labels are generated.
- **Generate preview:** The generated images and labels are visualized to verify the generation process.

### 4.3 Generator options

A number of arguments can be configured to control the generation process. Configuration of generator options is possible through command line GUI. Details regarding the arguments are provided in 1 and 2.

### 4.4 Sample results

Sample results of the artificial image generation algorithm can be seen in 10. The bounding box represents the object detection label and the different colors of the segmentation labels denote different label values.

| Generator options         | Description  |
|---------------------------|--|
| <b>image_dimension</b>    | Dimension of the real images.                                    |
| <b>num_scales</b>         | Number of scales including original object scale.                |
| <b>backgrounds_path</b>   | Path to directory where the background images are located.       |
| <b>image_path</b>         | Path to directory where real images are located.                 |
| <b>label_path</b>         | Path to directory where labels are located.                      |
| <b>real_img_type</b>      | The format of the real image.                                    |
| <b>min_obj_area</b>       | Minimum area in percentage allowed for an object in image space. |
| <b>max_obj_area</b>       | Maximum area in percentage allowed for an object in image space. |
| <b>save_label_preview</b> | Save image+label in single image for preview.                    |
| <b>save_obj_det_label</b> | Save object detection labels in csv files.                       |
| <b>save_mask</b>          | Save images showing the segmentation mask.                       |
| <b>image_save_path</b>    | Path where the generated artificial image needs to be saved.     |
| <b>label_save_path</b>    | Path where the generated segmentation label needs to be saved.   |
| <b>preview_save_path</b>  | Path where object detection labels needs to be saved.            |
| <b>obj_det_save_path</b>  | Path where object detection labels needs to be saved.            |
| <b>mask_save_path</b>     | Path where segmentation masks needs to be saved.                 |
| <b>start_index</b>        | Index from which image and label names should start.             |
| <b>name_format</b>        | The format for image file names.                                 |
| <b>remove_clutter</b>     | Remove images cluttered with objects.                            |
| <b>num_images</b>         | Number of artificial images to generate.                         |
| <b>max_objects</b>        | Maximum number of objects allowed in an image.                   |
| <b>num_regenerate</b>     | Number of regeneration attempts of removed details dict.         |
| <b>min_distance</b>       | Minimum pixel distance required between two objects.             |
| <b>max_occupied_area</b>  | Maximum object occupancy area allowed.                           |
| <b>scale_ranges</b>       | Can be used to change the zoom range of specific objects.        |

Table 1: Description of generator options

## 5 Splits of the dataset

## 6 Creation of dataset variants:

Different variants of the dataset are created based on the properties of the objects in the dataset, and the type of background images used for generation of artificial images.

| Generator options         | Default value | Is required?                           |
|---------------------------|---------------|--|
| <b>image_dimension</b>    | [480, 640]    | Not required                           |
| <b>num_scales</b>         | 'randomize'   | Not required                           |
| <b>backgrounds_path</b>   | -             | Required                               |
| <b>image_path</b>         | -             | Required                               |
| <b>label_path</b>         | -             | Required                               |
| <b>real_img_type</b>      | '.jpg'        | Not required                           |
| <b>min_obj_area</b>       | 20            | Not required                           |
| <b>max_obj_area</b>       | 70            | Not required                           |
| <b>save_label_preview</b> | False         | Not required                           |
| <b>save_obj_det_label</b> | False         | Not required                           |
| <b>save_mask</b>          | False         | Not required                           |
| <b>image_save_path</b>    | -             | Required                               |
| <b>label_save_path</b>    | -             | Required                               |
| <b>preview_save_path</b>  | None          | Required if save_label_preview is True |
| <b>obj_det_save_path</b>  | None          | Required if save_obj_det_label is True |
| <b>mask_save_path</b>     | None          | Required if save_mask is True          |
| <b>start_index</b>        | 0             | Not required                           |
| <b>name_format</b>        | '%05d'        | Not required                           |
| <b>remove_clutter</b>     | True          | Not required                           |
| <b>num_images</b>         | 20            | Not required                           |
| <b>max_objects</b>        | 10            | Not required                           |
| <b>num_regenerate</b>     | 100           | Not required                           |
| <b>min_distance</b>       | 100           | Not required                           |
| <b>max_occupied_area</b>  | 0.8           | Not required                           |
| <b>scale_ranges</b>       | None          | Not required                           |

Table 2: Default value of generator options and whether the options are required to be set.

## 6.1 Motivation

Looking into the objects present in the dataset, it is apparent that some objects are similar in certain aspects. For instance, the objects m20 and m30 are very similar to each other except that m30 is bigger in size and has a slightly different color. Because of the similarities existing among objects, the segmentation model could face certain difficulties as listed below:

- **Inability to distinguish size:** The segmentation model is given no information regarding the positions of the camera or the object in the real world. If camera extrinsic calibration information is available to the segmentation model, the model could possibly learn to distinguish different sizes. However, such information is

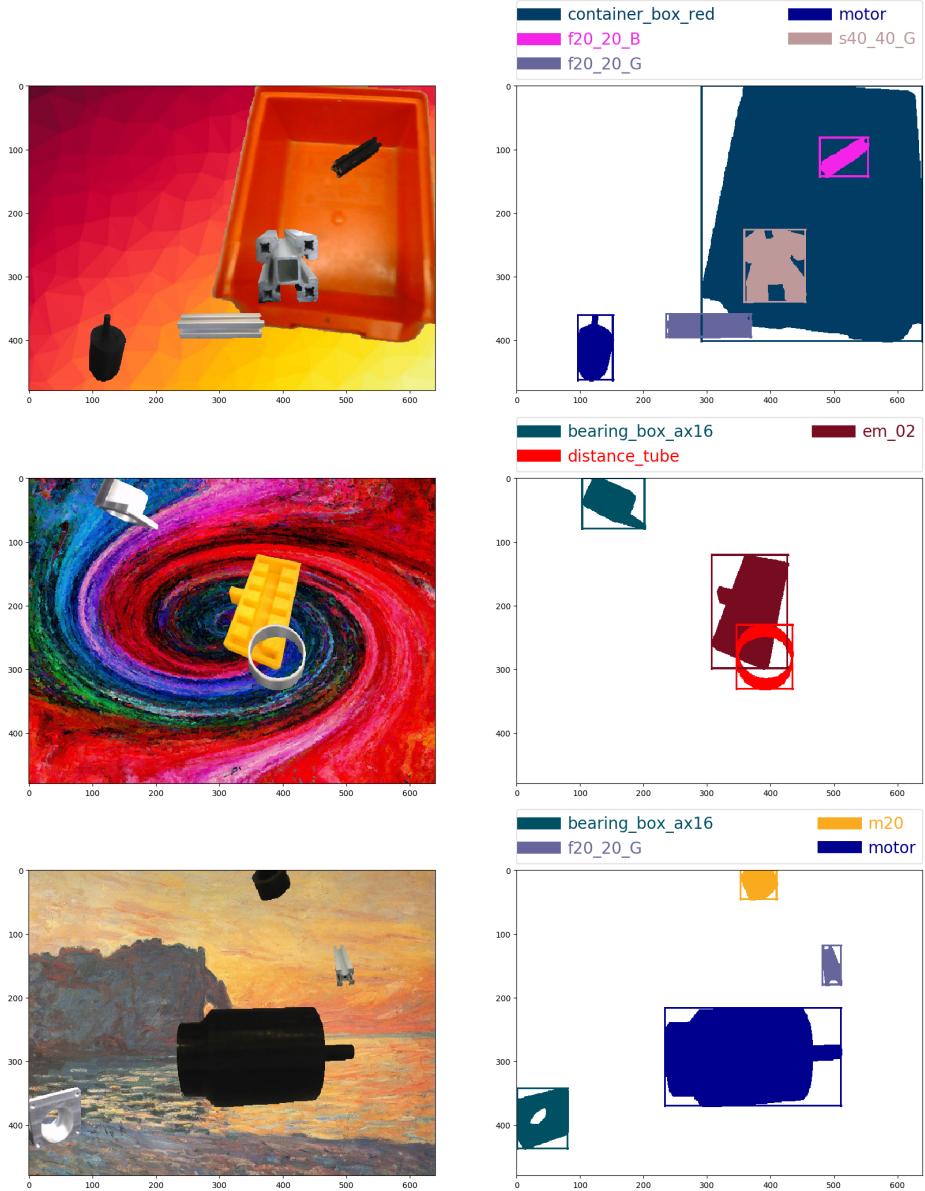


Figure 7: Sample results produced by the artificial image generation algorithm. In each row, the image on the left shows the generated artificial image and the image on the right shows a visualization of the semantic segmentation label and object detection label. At the top of every label visualization image, the objects in the image and their corresponding colors in the visualization are indicated.

not available. In addition, the objects in the artificial images are randomly scaled to different sizes thereby removing any size related information available.

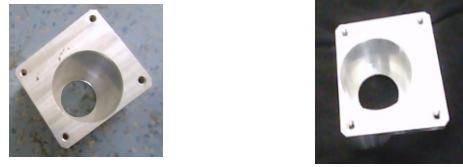
- **Inability to distinguish subtle variations in color:** The real images were taken under different lighting conditions. As a result, there is no consistent difference in color information available between classes. This makes it difficult for the segmentation model to learn patterns in color information.

- **Inability to distinguish shapes:** Certain objects are closely related to each other in terms of shape and differ only slightly. For instance, bearing\_box\_ax16 and bearing\_box\_ax01 are similar in shape except in a few viewpoints as illustrated in 8 and 9. In such cases, in certain viewpoints, the segmentation model would not be able to distinguish between similarly shaped objects.



(a) A viewpoint of bearing box ax01      (b) A viewpoint of bearing box ax16

Figure 8: A similar viewpoint of bearing box ax01 and bearing box ax16 where the difference in shapes between the two objects is clearly visible.



(a) A viewpoint of bearing box ax01      (b) A viewpoint of bearing box ax16

Figure 9: A similar viewpoint of bearing box ax01 and bearing box ax16 where they appear similar in shape.

## 6.2 Dataset variants

Four different dataset variants have been created as listed below:

- **atWork\_full:** This variant has different label values for all the 18 objects in the dataset and an additional label value for background. The total number of classes is 19 including background and the label values range from 0 to 18. The different classes in this variant along with their label values are listed in 3.
- **atWork\_size\_invariant:** In this variant, objects which are similar to each other in terms of shape but differ in size are combined together into one class. On this regard, f20\_20\_B and s40\_40\_B are combined and named f\_s20\_40\_20\_40\_B. Similarly, f20\_20\_G and s40\_40\_G are combined and named f\_s20\_40\_20\_40\_G. m20 and m30 are combined and named m20\_30. The two bearing boxes, bearing\_box\_ax01 and bearing\_box\_ax16 are also combined together as they are similar to each other in certain viewpoints. They form the new class bearing\_box.

The objects in this variant along with their label values are listed in 7. This variant is named "atWork\_size\_invariant" as in this variant, the major change deals with the ignorance of the size of the objects as distinguishing information.

- **atWork\_similar\_shapes:** In the previous variant "atWork\_size\_invariant", objects similar in terms of shape but different in terms of color were treated as separate classes. In this variant, variation in terms of color is also ignored. In addition to the previous variant, f\_s20\_40\_20\_40\_B and f\_s20\_40\_20\_40\_G are combined and named f\_s20\_40\_20\_40\_B\_G. The container boxes, container\_box\_red and container\_box\_blue were also combined to form the new class container\_box. This variant is named "atWork\_similar\_shapes" as objects with similar shapes are given equal label values. Details regarding this variant are listed in 5.
- **atWork\_binary:** An interesting question would be, "how would a segmentation model perform when it is just tasked with segmenting foreground from background". To address this question, an additional variant is created called "atWork\_binary" where all the objects of interest are combined to form the "foreground" class with label value 1. The "background" class retains its label value of 0. The classes in this variant are listed in 6.

| Variant Name | Object Name        | Label Value |
|--------------|--------------------|-------------|
| atWork_full  | background         | 0           |
|              | f20_20_B           | 1           |
|              | s40_40_B           | 2           |
|              | f20_20_G           | 3           |
|              | s40_40_G           | 4           |
|              | m20_100            | 5           |
|              | m20                | 6           |
|              | m30                | 7           |
|              | r20                | 8           |
|              | bearing_box_ax01   | 9           |
|              | bearing            | 10          |
|              | axis               | 11          |
|              | distance_tube      | 12          |
|              | motor              | 13          |
|              | container_box_blue | 14          |
|              | container_box_red  | 15          |
|              | bearing_box_ax16   | 16          |
|              | em_01              | 17          |
|              | em_02              | 18          |

Table 3: Details of the "atWork\_full" variant

| Variant Name          | Object Name        | Label Value |
|-----------------------|--------------------|-------------|
| atWork_size_invariant | background         | 0           |
|                       | f_s20_40_20_40_B   | 1           |
|                       | f_s20_40_20_40_G   | 2           |
|                       | m20_100            | 3           |
|                       | m20_30             | 4           |
|                       | r20                | 5           |
|                       | bearing_box        | 6           |
|                       | bearing            | 7           |
|                       | axis               | 8           |
|                       | distance_tube      | 9           |
|                       | motor              | 10          |
|                       | container_box_blue | 11          |
|                       | container_box_red  | 12          |
|                       | em_01              | 13          |
|                       | em_02              | 14          |

Table 4: Details of the "atWork\_size\_invariant" variant

| Variant Name          | Object Name        | Label Value |
|-----------------------|--------------------|-------------|
| atWork_similar_shapes | background         | 0           |
|                       | f_s20_40_20_40_B_G | 1           |
|                       | m20_100            | 2           |
|                       | m20_30             | 3           |
|                       | r20                | 4           |
|                       | bearing_box        | 5           |
|                       | bearing            | 6           |
|                       | axis               | 7           |
|                       | distance_tube      | 8           |
|                       | motor              | 9           |
|                       | container          | 10          |
|                       | em_01              | 11          |
|                       | em_02              | 12          |

Table 5: Details of the "atWork\_similar\_shapes" variant

| Variant Name  | Object Name | Label Value |
|---------------|-------------|-------------|
| atWork_binary | background  | 0           |
|               | foreground  | 1           |

Table 6: Details of the "atWork\_binary" variant

### 6.3 The shades of white dataset

The backgrounds used for the artificial image generation process are images with a variety of different colors, textures and so on. In essence, the background images do not seem to follow any pattern as such. As a result, the generated artificial images are un-

likely to be similar to an image taken by an atWork robot. In order to address this, the background images used in the artificial image generation process are all replaced with images which mostly contain shades of white color in them. With this as the only change, the entire artificial image generation process and variant creation process is repeated to arrive at a new dataset which is named "shades of white".

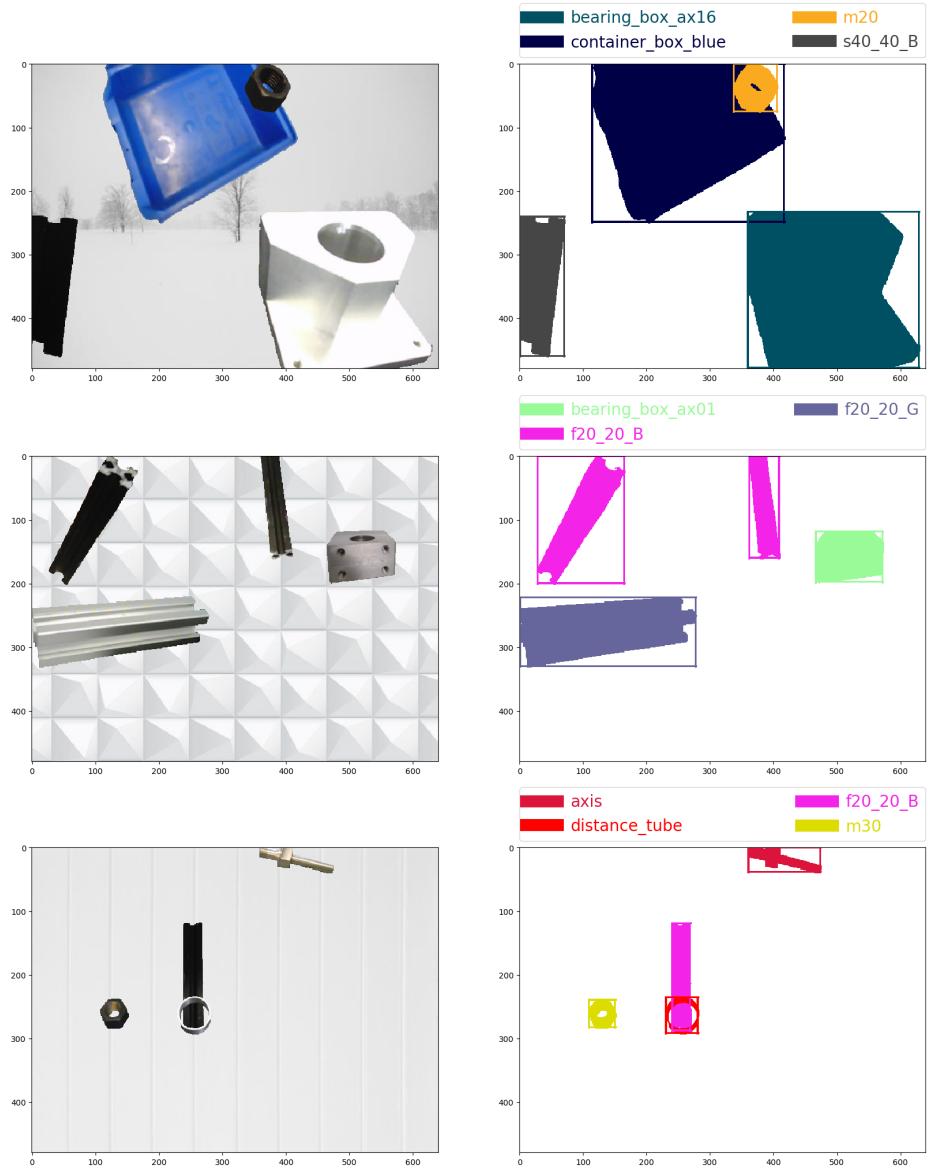


Figure 10: Sample results produced by the artificial image generation algorithm for the "shades of white" dataset.

## 7 Data analysis:

All the variants of the dataset are analyzed in terms of the pixels occupied by each class in percentage and the number of images in which each class appears.

### 7.1 Analysing the "atWork\_full" variant:

## 8 Meta-data of the dataset

Details regarding the dataset is provided in table 3.

|                  | Training                                      | Validation                                 | Test                                       |
|------------------|---|--|--|
| Real Images      | 30 per object.<br>Total: $30 \times 18 = 540$ | 5 per object.<br>Total: $5 \times 18 = 90$ | 5 per object.<br>Total: $5 \times 18 = 90$ |
| Augmented Images | 5000  | 810  | 810  |
| Total Images     | 5540  | 900  | 900  |

Table 7: Meta-data of the dataset

## 9 Notable features of the artificial image generator

## 10 Conclusion and possible directions of improvement

Creating a custom dataset for a desired application is evidently challenging. To overcome the time consuming nature of creating annotations for semantic segmentation, choices such as 1. placing just 1 object per image while taking real images and 2. augmenting the objects on a random selection of diverse backgrounds, were made. This method of augmentation, although inspired by dataset generation method used in [2] and the Synthia dataset [3], takes a different approach. Unlike [2], which uses 3D CAD models, this approach does not require any 3D models. Also, this approach does not require a virtual world as used by the Synthia dataset [3]. However, the effectiveness of the dataset is yet to be verified by training and testing available segmentation models. The following list provides possible directions of improvement:

- The ImageLabeler app by default saves the label '.png' file with the name 'Label\_1.png' in a folder called PixelLabelData. A automation script can be written and added to the ImageLabeler to provide options to save the label file in a way the user wants.

- Creating a way to replace all unlabeled pixels with the label value of 'background' from within the ImageLabeler would be helpful. For now, this is done by first exporting the label, then loading the label using opencv in python to replace 0 (value of unlabeled pixels) with 19 (value of 'background').
- The augmentation script is written in python and is independent of the MATLAB ImageLabeler app. This can be improved by including a way to start augmentation right from the ImageLabeler.
- A GUI for the augmentation script can be created to improve the ease of use.

## References

- [1] github: robocup-at-work/rulebook/images; link: <https://github.com/robocup-at-work/rulebook/tree/master/images>.
- [2] Param S. Rajpura, Manik Goyal, Hristo Bojinov, and Ravi S. Hegde. Dataset augmentation with synthetic images improves semantic segmentation. *CoRR*, abs/1709.00849, 2017.
- [3] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio Lopez. The SYNTHIA Dataset: A large collection of synthetic images for semantic segmentation of urban scenes. 2016.