# Synthetic Data Generation for Mask R-CNN to Estimate Water Absorption in Cement Pastes

Tony Joo Ahmed Ibrahim Jordan Wu Hossein Kabir

## An Updated Statement of the Project Definition and Goals

This project aims to leverage Mask RCNN to continuously estimate the water absorption level based on synthetic, or real images, or a combination of both these datasets. We did not change the scope of our research, and we did our best to stick to our original plan. In addition, we have created a Github page which is publicly available and can be accessed using the URL below that provides a more detailed description of the project and the theory behind it.

tonyjoo974/mask-rcnn-using-synthetic-models: Synthetic data generation for Mask R-CNN enabled water absorption estimation in cement pastes. (github.com)

## Current member roles and collaboration strategy

*Tony Joo* and *Ahmed Ibrahim*: Manually annotating natural/ real images (images *001a.jpg* to *901a.jpg*, and *001b.jpg* to *901b.jpg*, respectively, which can be accessed from the real.zip dataset) , optimizing the employed Mask RCNN algorithm to be compatible with our dataset, and editing results on Github .

*Jordan Wu*: Manually annotating natural/ real images (images *001c.jpg* to *901c.jpg*, which can be accessed in real.zip dataset), optimizing the employed Mask RCNN algorithm to be compatible with our dataset, and developing an algorithm that replaces CVAT image annotation tool using flood_fill_helper.ipynb.

*Hossein Kabir*: Manually annotating natural/real images (images *001d.jpg* to *901d.jpg,* which can be accessed in real.zip dataset), creating synthetic.zip image dataset (images *aSlide1.jpg* to *dSlide225.jpg*), optimizing the employed Mask RCNN algorithm to be compatible with our dataset, and defining the "water" class WaterDataset_with_its_utils.py for detecting and labelling the water level.

## Proposed approaches

### 1) Manual Annotation of Real Images

We realized that manual annotation of labels via CVAT is very unergonomic and time-consuming for our sample dataset. Therefore, we utilized the fact that for every frame of each specimen the borders of the subject are constant, and that only the height of the water varies. As a result, we developed an algorithm (flood_fill_helper.ipynb) to annotate the dataset that reads the data from the stylus of any tablet (e.g., Ipad) to facilitate image annotation. Detailed explanation of this method can be found on Github.

### 2) Synthetic Generation of Images

Our original plan was to leverage Rhinoceros software for creating synthetic image dataset; however, since it is not an open source software, we decided to simply use Microsoft Powerpoint to easily and rapidly create synthetic images as well as their corresponding masks. In the next step we manipulated (binarized, renamed, and resized) and created our dataset using binarizing_and_resizing_images.py such that it would be compatible with our Mask-RCNN model. Again, a detailed explanation on how we created the synthetic image dataset is available on our Github.

## Implementing Mask- RCNN

Mask R-CNN is built upon Faster R-CNN, which proposes a method for real-time object detection with Region Proposal Networks (RPN) **[1]**. This method is a 2-stage framework, where the first stage extracts features by passing the image through a CNN backbone architecture based on ResNet (ResNet101 in this study), which outputs feature maps **[2]**. Then, using the feature maps, RPN proposes regions by using sliding window technique on k anchor boxes. The second stage then extracts features from each candidate box using RoIAlign and it performs classification, bounding-box regression, and outputs binary masks for each RoI "in parallel" to be used for instance segmentation. We also used feature activation output in the last residual stage of each stage to improve the robustness of lower-level feature extraction. Besides, the number of classes defined for our model is 2, i.e., including 1 class for background, and 1 class for "water". The model is trained based on 20 epochs (300 iterations per epoch) with an initial learning rate of 0.001, in which the learning rate is later decreased by a factor of 10 for every 5 epochs. In addition, the model is established based on TensorFlow (version 1.15.2) and Keras (version 2.3.1) libraries, which are found to be compatible with our model. It is worth noting that the training procedure was done using NVIDIA T4 GPUs with 48 GB DDR5X memory (which can be accessed in Google Colab) and is based on the original implementation, which is currently under the MIT license **[3]**. Moreover, roughly 90% of the images were allocated to our training dataset, and the rest 10% were designated to the testing dataset. Finally, a detection threshold, i.e., intersection-over-union (IoU), of greater than 95% is selected, meaning that the proposals with confidence of less than 0.95 is not considered to be true positive.

## Data

Within the past 30 days, we have manually annotated 3604 real images (i.e., 901 images by each team member). These images are frames from video recordings of water absorptions by real cement pastes. **Fig. 1** is an example of the image and its annotation. Specifically, as shown in **Fig. 1**, attempts have been made to generalize our model by including images that have both specular (left column) and diffuse (right column) reflections. Similarly we have created 810 synthetic images and annotations, and again, it was tried to design both specular and diffuse to better generalize our model, and to make it less sensitive to various illumination conditions.

## Initial Results

Considering **Fig. 2**, it can be realized that the employed algorithm (trained based on synthetic data) is capable of marking the water level, even if it is applied on a video. Specifically, this figure shows the robustness of the employed method that can mark the variation of water level with time. However, the subplots shown in **Fig. 2** represent an easy dataset (without specularities) and as a result it is of interest to determine the performance of this method for analyzing difficult and complex specular images.

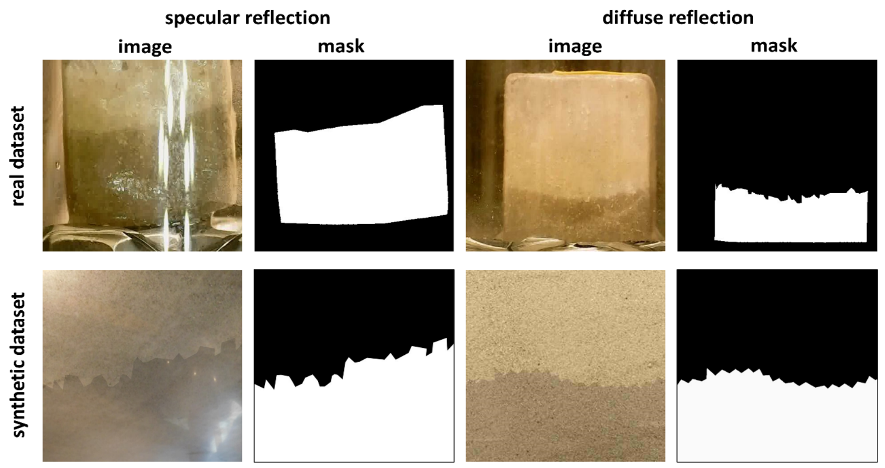

**Fig. 1**: Creating real (natural) and synthetic image dataset such that they have both specular and diffuse reflections.

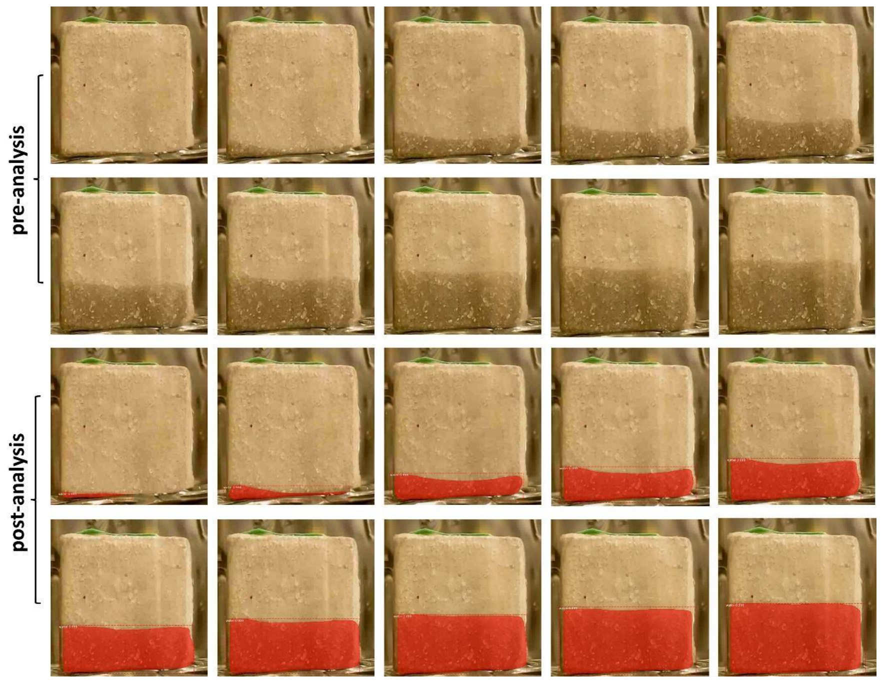

**Fig. 2**: Pre-analysis (top 2 rows) and post-analysis (bottom 2 rows) of a video using the developed Mask R-CNN.

As it was previously observed, the present Mask RCNN model (trained on synthetic dataset) is effective for analysis of simple images. However, as shown in **Fig. 3a**, we realize that our model can analyze complex images <u>if the synthetic and real image datasets are merged</u>. As a result, followed by merging the two synthetic and real dataset, we realized that our model is more accurate for analyzing difficult images, see **Fig. 3b**. However, our model still fails to delineate the region of interest for super difficult images (last row of **Fig. 3b**) specifically if the water level is high, which is a limitation of our method, but we will try to address it in the next two weeks.

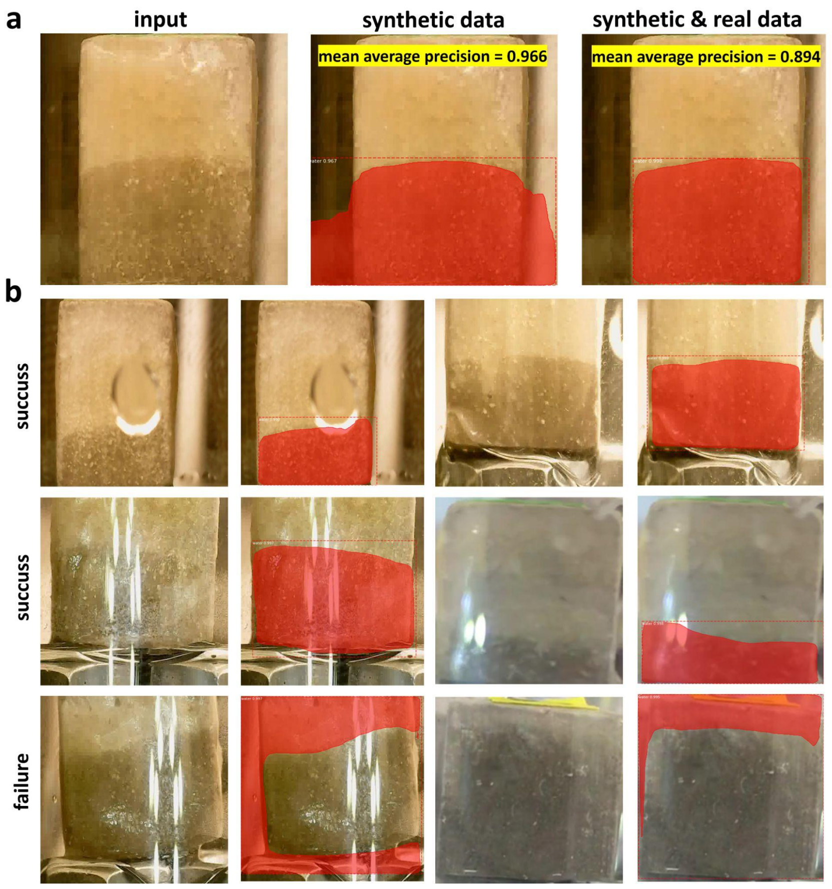

**Fig. 3**: (a) training the algorithm based on synthetic dataset, (b) based on a combination synthetic and real dataset.

## Current Reservations and Questions

So far we have realized that our model can accurately mark the region of interest (water level) even on complex images, but we still need to model more specular synthetic images to further improve the accuracy of our model. Furthermore, we also realized that our model gives incorrect estimations whenever the water level is high. As a result, in the remaining time (next two weeks) we will dedicate our time to create models which have relatively high water levels. Besides, if time permits, we want to possibly take some time to investigate other off-the-shelf segmentation models that are less complex compared to Mask RCNN, such as U-Net. We would also appreciate your comments and suggestions, which will surely help us improve further the quality of our research.

## References

**[1]** He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 2961-2969).
**[2]** He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
**[3]** https://github.com/matterport/Mask_RCNN.git