2020 CORSMAL Challenge - Team NTNU-ERC Report

Guilherme Christmann[1] and Jyun-Ting Song[1]

Dept. of Electrical Engineering
National Taiwan Normal University
Taipei, Taiwan
guichristmann@gmail.com

Abstract. This report presents the solutions of team NTNU-ERC for the 2020 CORSMAL Challenge. The challenge consisted in using multimodal data including depth, RGB, infrared, audio and IMU data to perform three tasks that involved containers. For the task of classification of filling type we utilized MFCC features extracted from audio as input a convolution-based model. In the task of estimating the volume of containers, we used image processing to determine the region that included the container. By feeding the region together with its dimensions to a convolution-based model we can estimate the volume of the container. Our team got first place in the task of estimating container volume. The full implementation is available at https://shorturl.at/CFKWZ.

1 Introduction

This technical report details the solutions of team NTNU-ERC for the 2020 CORSMAL Challenge. The challenge consisted of three tasks that involved different types of containers such as glasses, cups and boxes. The containers were either filled with water, rice or pasta, or left empty. The overall goal was to estimate the weight of the container, including its filling. This is a necessary skill for robots to robustly perform handover interactions with humans. Failure to estimate the proper weight of a container could result in excessive or insufficient grasping force, which could lead to spilling its contents or even damaging the container.

The three tasks were:

1. Fullness classification: is the container empty, half-filled (50%) or near-full (90%)?
2. Filling type classification: what was the container filled with: nothing, water, rice or pasta?
3. Container capacity estimation: what is the maximum capacity, i.e. volume, of the container?

The predictions for each task could be performed by an independent model, and the solutions are merged to get the final prediction of the overall task. Due

to timing constraints our team could only realize solutions for tasks 2 and 3, that is, classification of filling type and estimation of container capacity, respectively. The rest of this document details our methodology and results for each of these tasks.

2 Methodology

This section describes the methods utilized for each of the tasks, including data pre-processing, normalization, feature extraction, model definition, hyperparameters and training. We also briefly present the dataset, and our development environment.

2.1 The CORSMAL Dataset

The dataset has 1140 instances of multi-modal data recordings of people interacting with 15 different containers [1]. The interactions consisted of people manipulating the containers, such as pouring water into a cup, or shaking one of the box containers. The dataset was split into three different sets. The training set which included 9 containers and accompanied annotations; the public test set which included 3 containers and no annotations; and the private testing set which included the remaining three containers and its data was only accessible to the competition organizers.

The dataset was recorded using RGB and depth cameras (Intel RealSense D435i), as well as microphone arrays. For every sample in the datasets the available modalities were the following: 4 distinct RGB, infrared and depth perspectives, audio and IMU data. The four perspectives were first-person from human point-of-view, first-person from robot point-of-view, and two lateral perspectives.

It is important to note that the dataset was quite challenging, and it's not likely that a good solution can come from using just one modality. For example, there are scenarios where the person manipulates the container outside the field-of-view of all cameras. Furthermore, among the samples, there are very short and very large sequences, variations in lighting, scenery changes, noise from other environments, and changes in placement of objects as well as inclusion of non-related objects such as a jug, etc..

2.2 Development Environment

We downloaded the almost 400 GB of data and developed our solutions locally. The relevant specifications and software utilized are presented on Table 1 below.

CPU	Intel i7-8700 3.2GHz – 6 Cores
GPU	1x 1080 Ti
RAM	32 GB
DL Framework	PyTorch 1.6.0
Image Processing	OpenCV 4.4.0
Audio Processing	librosa 0.8.0

Table 1. Computer specifications and relevant software

2.3 Task 2 – Filling Type Classification

Task 2 consisted in classifying the type of filling among four possibilities: nothing (empty container), rice, pasta or water. We approached this task as a classification problem.

For this task we only utilized the audio data. As previously mentioned in Section 2.1, video data does not provide reliable information for all samples. In some of the recordings the contents are poured into the containers from out of view. Furthermore, some of the containers are opaque, so it is not possible to visually observe its contents at all. On the other hand, audio data provides valuable information across all samples. It is possible to hear contents being poured even from outside of view, or the shaking sound in the case of totally opaque containers such as the food boxes.

Data Pre-Processing

The training set contained a total of 684 audio recordings which were sampled at 22 kHz. We performed a random validation split of 10%, resulting in 615 recordings for training and 69 recordings for validation.

Our feature extraction consisted of computing the Mel-frequency cepstrum coefficients (MFCCs) for each audio recording [2]. We used a window size of 20 milliseconds and 40 MFCCs per window. Because the audio recordings had varying length we defined a maximum length of 30 seconds and zero-padded shorter recordings to this length. We normalized each MFCC by subtracting the mean and dividing the standard deviation over the whole sequence. Our resulting tensor shape was 1501 timesteps by 40 features for each audio recording.

The dataset had an unbalanced number of samples per class: 108 for nothing, 216 for pasta, 216 for rice, and 144 for water. This can become a large problem specially considering the small amount of samples overall. To prevent the model from biasing towards classes with more samples, we computed weights inversely proportional to the number of samples in each class.

Model Definition

We defined a simple convolution-based model where the input is the 1501x40 MFCC sequence. In this regard, we can imagine the sequence as if it was an "image". Our model is quite small, with only two convolution layers followed by

the fully-connected (output) layer with softmax activation. The first convolution layer has 16 filters, stride of 4 and a kernel of size 120x40. The first dimension of 120 enables the layer to process a large sequence of the input, and the second dimension of 40 ensures that all MFCCs contribute at each convolution operation. This is followed by the second convolution layer with 8 filters, stride of 2 and kernel size of 40x1. We also applied dropout of 25% in the output of the convolution layers. Figure 1 presents a simplified schematic of the model.

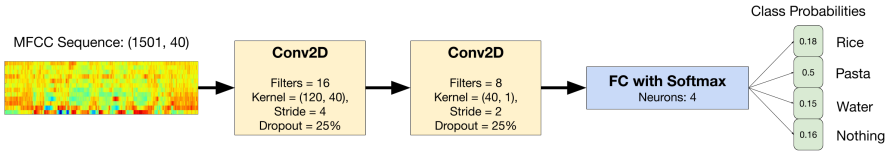

Fig. 1. Schematic of convolution-based model utilized for Task 2. The input are MFCCs sequences from audio recordings, and output one of the four filling type classes.

The model had approximately 86 thousand parameters. We employed SGD optimizer [3] with a fixed learning rate of 0.00025 and momentum of 0.9, cross-entropy loss taking into account the class weights, and used a batch size of 16. The model was trained for 200 epochs, which takes less than 1 minute of real-time in our development environment (see Section 2.2).

2.4 Task 3 – Container capacity estimation

Task 3 consisted in estimating the maximum capacity, i.e. total volume, of the containers in each recording. We approached this task as a regression problem with a single output: the container volume.

Prior to estimating the actual container volume it is necessary to locate its position in the data. Because there is no annotations provided for the locations of the container in the image, and due to the fact that there's a lot of variation between the recordings, this step becomes quite challenging. One approach would be to use segmentation or object detection models to find the container in the different modalities and fuse the information of synchronous frames. However, in part due to timing constraints, we opted for a simpler approach, using a single modality: the depth camera from the robot's perspective.

Even though there's a lot of variation between different recordings, there is one event that is consistent throughout all of them: the human always presents the container to the robot towards the end of the recording. We designed a method based on finding the closest contours to the camera that can reliably find the region of interest (ROI) of the container. We start at the very last frame of the sequence and iterate backwards until a strong candidate is found. Since sometimes the human puts the container down before the end of the sequence, it is not guaranteed that the last frame will provide a reliable ROI. Our method for finding a strong bounding box is as follows:

1. Discard the bottom 16% of the image to disregard the table as shown in Figure 2. This eliminates false positives from the region of the table that is close to the camera. Since the humans present the container holding it in the air this is a safe assumption to make;
2. Discard all depth pixels which value is larger than 700. This corresponds to a distance of 70 centimeters. Shown on Figure 3;
3. Apply morphology operation of closing with a 15x15 kernel to "fill" some of the spots in the remaining image, shown in Figure 4;
4. Find large contours (pixel area ≥ 15000), as shown in Figure 5. If more than one contour is found, choose the one whose area is closest to the camera (smaller pixel values);
5. Compute the ROI bounding box of the contour and expand it by 5%. The final extracted ROI is presented in Figure 6.

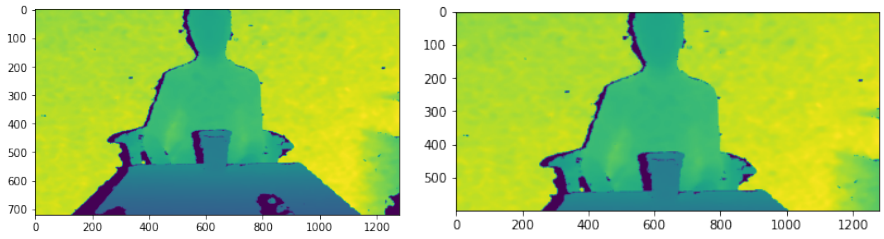

Fig. 2. Left: original depth image. Right: discarded the bottom 16% of the image.

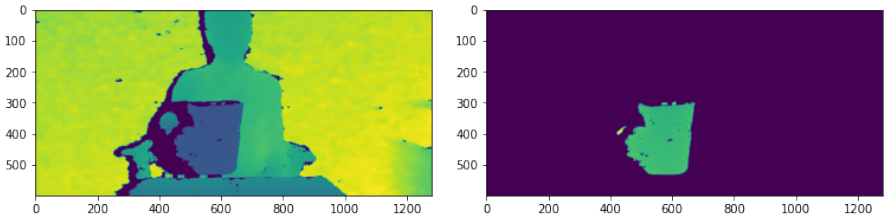

Fig. 3. Before and after filtering. Pixel whose distance is more than 70 centimeters are discarded.

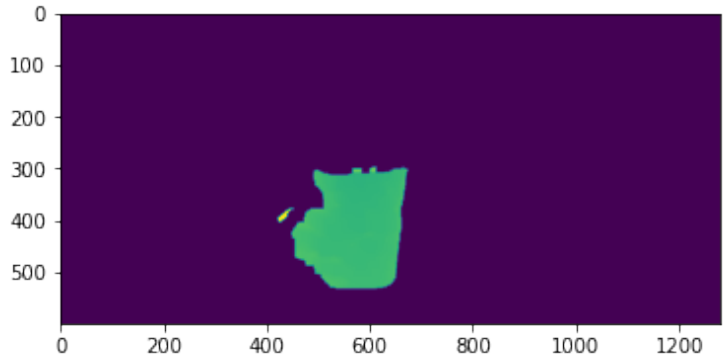

Fig. 4. Morphology operation to close gaps in the filtered image.

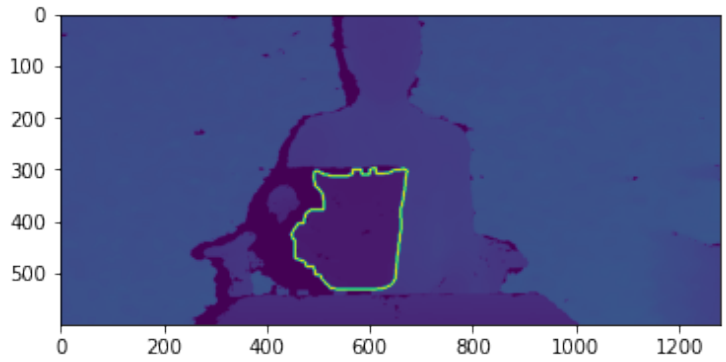

Fig. 5. Detected contour.

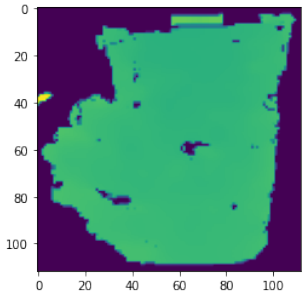

Fig. 6. The final extracted ROI of the container.

Data Pre-Processing

The dataset is constructed by extracting the ROIs from the sequences in the training set. Out of the 684 sequences in the training set, our method failed to find a ROI in 12 of them, which were then discarded from training. The ROIs were resized to 112x112 and normalized by the maximum pixel value of 700. In addition, the ROI dimensions are also an input to the model and are normalized to be between 0 and 1, relative to the original dimensions of the image. The training targets, i.e. volume of the containers, are normalized dividing by 4000. With this normalization, the maximum possible volume that the model can output is 4000 mL. We employed a 15% validation split, which resulted in 572 sequences for training and 100 sequences for validation.

Model Definition

For this task, we also constructed a convolution-based model, a few layers deeper than the one used in Task 2. Our model has two inputs, a depth image with the extracted container of size 112x112, and the original dimensions of the extracted ROI relative to the image. Figure 7 presents a schematic of the model.

The model consists of four convolution layers and three fully-connected layers. The depth ROI is processed by the convolution layers, and its outputting features are concatenated with the second input, the ROI dimensions. The concatenated features are then passed through another fully-connected layer with a single output that encodes the estimated container volume. Every convolution block consists of a 2D convolution layer with ReLU activation, followed by batch normalization [4] and max pooling with a stride of 2. The model had a total of 532 thousand parameters.

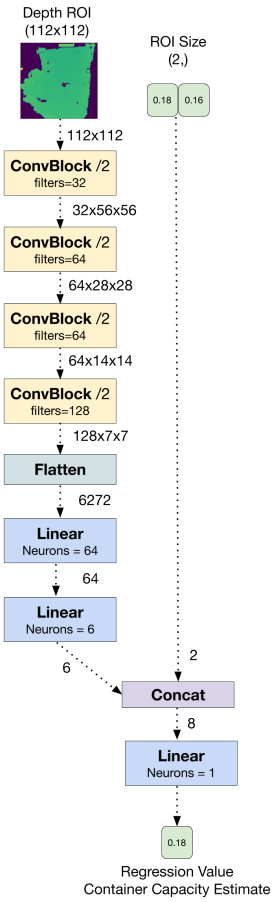

Fig. 7. Schematic of the model used for Task 3. The inputs are the extracted ROI from the depth image with the container and its dimensions. The output is the estimated volume of the container.

The optimizer used was Adam, with a learning rate of 0.00025 with minimum squared error as the loss function. The model was trained with a batch size of 8 for 200 epochs, which took less than 2 minutes of real-time in our configuration (see Section 2.2).

3 Results

The official table of results can be accessed at `https://shorturl.at/bitJQ`. Overall, our results beat the random baseline. However, we didn't realize the task of estimation of the filling level, and only placed 4th overall. We got 38.56

points in the public testing set, 39.80 points in the private testing set, and 39.18 points overall.

We also placed 4th in Task 2, classification of the filling type, reaching an accuracy of 81.97% in the public testing set, and 91.67% in the private testing set. Our overall accuracy was 86.89%.

Most notably, even though we were using a quite simple method, we placed first in Task 3, estimation of the container volumes. We got 66.92 points in the public testing set, 67.67 points in the private testing set and 67.30 points overall.

4 Conclusion

In this report, we presented the solutions of team NTNU-ERC for the 2020 CORSMAL Challenge. The challenge consisted of three different tasks involving multi-modal data of different types of containers. The dataset had quite a bit of variance among its sequences, and several different scenarios. Due to timing constraints, we only performed two of the three tasks. In the task of classifying the type of container we employed a small convolution-based network to process MFCCs features extracted from audio data. And, in the task of estimating the volume of the containers, we used image processing to reliably find the container region in a depth sequence, and used a larger convolution network to perform the estimates.

Most notably, we got first place in the volume estimation task, even though our method was quite simple. Our methods were quite lightweight, and our models can be trained on a single GPU in under 5 minutes. That being said, our solutions could be improved by using more complex models such as shape estimation models. Additionally, using segmentation and object detection models to locate the containers would provide more robust solutions, that could make use of more modalities.

5 Acknowledgements

We would like to express our thanks to the organizers and panel of judges of the 2020 CORSMAL Challenge and 2020 Intelligent Sensing Summer School for these opportunities. They were supportive throughout all of the events and provided valuable learning resources during the Summer School.

References

1. A. Xompero, R. Sanchez-Matilla, R. Mazzon, and A. Cavallaro, "Corsmal containers manipulation," 2020.
2. M. R. Hasan, M. Jamil, M. Rahman, *et al.*, "Speaker identification using mel frequency cepstral coefficients," *variations*, vol. 1, no. 4, 2004.
3. L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*, pp. 177–186, Springer, 2010.
4. S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.