Container Number Recognition using AI for planning at R&D Yard

---

1. Problem Statement

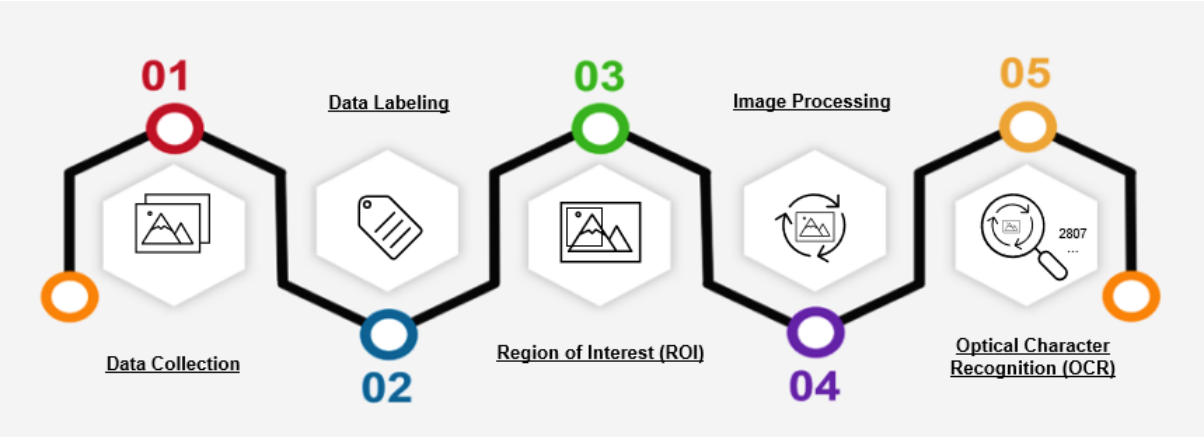
It is largely essential for port authorities to capture important information on containers such as the container number, ISO number, seals, license plate, etc. to track the container flow. Currently, in most places, the data associated with these containers are read manually without resorting to any automation gadgets [1].

Inspectors may sometimes oversight these containers causing reading errors in places with insufficient lighting, distance or other such factors. Human errors causing misread entities and omissions can be costly and thus automating the entire process through machines can be a viable solution.

Through computer vision, Optical Character Recognition (OCR) models can be deployed that can leverage existing hardware, and provide automated texts recognition without any human intervention.With these models, we can capture all the information that we need.

2. Proposed Approach

The aforementioned use-case will be dealt with a five-step process comprising of data collection, data labeling, marking the Region of Interest (ROI), image processing, and finally an Optical Character Recognition model to read the numbers and letters in the processed images. The figure below demonstrates the overall workflow of this project.



2.1. Data Collection

The data to be collected are the raw images or image frames of a live video footage of container doors, captured by mounted cameras, drones, etc. The image below is representing a sample door of a container marked by relevant information.

2.2. Data Labeling

Using relevant image labeling tools like the labeling, the area of information on the container is labeled for automating the identification process for finding the region of interest.

2.3. Marking the Region of Interest (ROI)

Using custom detection AI models such as the state of the art YOLO model, we train the labeled images for finding the regions of interest and using computer vision, we crop the detected region for further processing.

The image on the left is the marked ROI, and right, we have the cropped image.

2.4. Image Processing

The cropped region of interest is further processed for better readability and identification of texts in the image. We convert the RGB image to binary and with proper thresholding, we transform the image to retain only relevant parts of the image, which is the container number. The image on the left shows the ROI and the right, we have the image after transformation.

2.5. OCR Model

The processed image is sent to the OCR model to retrieve the text information in the image.


```
Owner Code          :  CBC
Product Group Code  :  U
Registration No.    :  200031
Check Digit         :  7
ISO Code            :  22G1
```

3. Proposed Models

3.1. Detecting the Region of Interest (ROI)

For the purpose of detecting the information on the doors of the container, the images are manually labeled using labeling tool to get the annotations that detail the coordinates of the bounding boxes in the image, for training. The model used here is the DarkNet YOLOv3 object detection model. YOLO (You Only Look Once) models are popular for fast object detection, developed by Joseph Redmon, et al. specifically designed to cater to real-time detection scenarios [2].

Although there's a slight performance hit in terms of accuracy, these models are popular because of their processing speed, often demonstrated in real-time on video or with camera feed input which match our real-world use-cases.

3.2. Optical Character Recognition (OCR)

For OCR, three different approaches can be taken,

1. Using existing free OCR engines that are directly compatible with python, for eg, PyTesseract
   Drawback – Not very accurate, unreliable for real world use cases.
2. Building a custom OCR model with Keras, TensorFlow and Deep Learning.
   The model will be trained on MNIST datasets for recognizing numbers 0-9 and alphabets A-Z using ResNet's architecture to model the recognition. *(Yet to implement and test)*
3. Using Public APIs for OCR such as Nanonets, Azure Computer Vision API, Google's cloud vision API.
   Drawback – Cost, local and cloud deployment, compatibility with our models.

4. Project Structure

```
|──custom_images
|     |──1.jpg, 2.jpg, …. (training images, annotations for Bounding Box Coordinates)
|     |──classes.NAMES
|     |──creating-files-data-and-name.py
|     |──creating-train-and-test-txt-files.py
|──OCR_img
|     |──info.jpg
|     |──processed.jpg
|──yolo_model_cfg
|     |──yolov3_custom.cfg
|──yolo_model_weights
|     |──yolov3_custom_final.weights
|──custom.py
|──ocr.py
```

custom_images contain the images for training the model along with labeled bounding box annotations, classes.NAMES file for pretrained YOLO model for finding ROI.
creating-files-data-and-name.py and creating-train-and-test-txt-files.py are used for splitting the images and annotations files into train and test sets.

OCR_img contains the cropped ROI and transformed image to be fitted to OCR model.

yolo_model_cfg and yolo_model_weights contain the custom YOLO model's configuration and weights file after training – to be used for testing on images.

custom.py is used for detection and main YOLO's algorithm with appropriate confidence thresholds.

ocr.py contains the ocr python script. *(under development)*

5. Project Requirements

- Cameras with live footages that are stable, clear images.
- Drones that can capture clear videos of containers.
- Cloud storage to deploy model pipelines in cloud.
- Ubuntu 16.04 or later (64-bit) Operating System
- macOS 10.12.6 (Sierra) or later (64-bit) (no GPU support)
- Windows 7 or later (64-bit) Operating System
- GPU support requires a CUDA®-enabled card (Ubuntu and Windows)

6. Challenges/Issues Faced

- Improper lighting conditions for images.
- Images at different angles that do not capture the information clearly.
- Images/videos captured in the night may diminish quality of images.
- ROI Detection model is not 100% accurate.
- OCR Model may not be accurate in all conditions.
- Single image/frame having multiple doors and regions of interests – May not detect all of them.
- The model may require the information on doors to be clear and vivid.
- Existing systems and public API's may outperform the proposed model.

7. Current Status

- Currently working on developing a custom OCR model using TensorFlow, Deep learning.

References

[1] https://nanonets.com/case-study/shipping-container

[2] https://machinelearningmastery.com/how-to-perform-object-detection-with-yolov3-in-keras/