Software Test Plan Draft

Team 5: LPR Correction

Purpose

The purpose of these tests is to test the functionality, accuracy, and efficiency of the object recognition and optical character recognition components within the LPR software. The tests follow the project requirements and test for high accuracy, fast performance, reliability, and the visibility and functionality of the user interface.

Scope

The tests focus on the three components of the solution: Object Recognition, Optical Character Recognition, and the User Interface. They primarily focus on the accuracy and run-times of the recognition components and the functionality of the User Interface. The testing could be expanded to incorporate negative testing and not just positive scenarios where the solution is expected to work. It could also contain tests to understand the behavior of the system with blurry images, incorrect plate orientations or more obstructed plates.

Table 1.0: Requirement/Verification Cross-Reference Matrix

Classes of Verification: I- First Article, II-Environmental, III-Acceptance Test, IV- None

Methods of Verification: A- Analysis, T-Test, D-Demonstration, I- Inspection

Project Requirements	Test Names	Verification Class			
		I	II	III	IV
High Accuracy	1.1 OR Accuracy clear images	T		T	
High Accuracy	1.2 OR Accuracy challenging images	T		T	
High Accuracy	1.3 OCR Accuracy	T	I	T	
Fast performance	2.1 OCR Run-time	T		T	
Fast performance	2.2 OR Run-time	T		T	
Fast performance	2.3 Combined OR-OCR Run-time	A	I	D	
Reliability	3.1 Image Loading Speed Variance	D		D	
User interface	4.0 UI Set-up	D	I	D	
User interface	4.1 UI Feature Visibility	D	I	D	

User interface	4.2 UI Navigation	D		D	

Testing Procedure

This section describes the scope of each test, overall procedure such as types of tests, main specification such as load, height, and dimension to be tested, and pass/fail criteria.

1.1 OR Accuracy with clear images

One of the main problems with CampusParc's existing LPR solution was difficulty reading plate information for non-traditional vehicles. That included vehicles where the plate is not in the center of the rear like Jeeps, commercial vehicles with words outside of the license plate, and vehicles with bumper stickers. To prevent LPR errors due to these factors, the recognition process was split into two steps. The first step uses the YOLOv8 Object Recognition algorithm to identify the vehicle and license plate in a still picture captured from the existing cameras. The solution must be able to correctly identify the license-plate in an image and then crop the image to include only the plate. Since the pictures are captured in different locations under various conditions, the clarity and location of the license plate are not always consistent. The presence of fog, glare, or dirty plates due to weather can also impact the LPR process. To ensure the solution can accurately read a license plate during all these conditions, several automated tests will be completed. First, the YOLO model will be tested on 1000 clear images (which are believed to be easily interpreted) using the YOLO "predict" function in a Google Colab Juptyer Notebook script. The accuracy will automatically be recorded identifying the two classes: vehicle and license-plate. For the test to pass, the model must see accuracy over 90% accurate.

Procedure:

1. Set up OR environment on Google CoLab
2. Set up the folder of all clear images of vehicles
3. Run the YOLOv8 model
4. Interpret the accuracy of recognizing the license plate (LP) on the vehicles

1.2 OR Accuracy with the challenging images

The next test will contain around 300 boundary images i.e. those with the challenging scenarios described above. The same procedure is followed as in test 1.1 and the test will be considered as a "pass" if its 85% accurate or higher. If either of these tests fail, the model will be retrained accordingly.

1.3 OCR Accuracy

A set of 1000 images will be sent through the Optical Character Recognition (OCR) component of the LPR software and the resulting LP data will be compared against the truth labels that have been manually prepared by the team. The test code will then calculate the accuracy based on the number of exact matches to the truth labels. If the calculated accuracy is not above 95%, the test will be considered a failed test.

Procedure:

1. Set up the OCR environment on Google Colab
2. Set up a folder with all the extracted good license plate
3. Create a truth.txt by reading the license plates manually
4. Run the folder on OCR
5. Run the OCR software
6. Calculate the average accuracy based on the confidence level provided for each plate
7. Compare the read plates with the truth.txt file created manually

2.1 OCR System Run-time

The system run time test involves testing a small random set of images to determine the average time it takes the software to process each image. The test will measure the total time it takes the OCR to process all the images and the time will be averaged to get an estimated time per image. The test will be run on multiple sets of random images that are the same size, and all the resulting average times will also be averaged to get a more precise measurement. The goal outlined by the requirements table is anywhere between 40 and 100 milliseconds. Splitting this into two parts for the OCR and OR, the test range becomes 20 to 50 milliseconds for each. A result that is more than 10% longer than this range will be considered a failure. When combined with the OCR test speed the result should be no longer than 1 second per image.

Procedure:

1. Set up OCR environment on Google Colab
2. Obtain a folder of images
3. In the test code file:
    a. Get the start time
    b. Call the OCR code with the image folder
    c. Get the end time
4. Subtract the end time from the start time to get the total run time
5. Repeat for multiple folders of images

2.2 OR System Run-time

Like the OCR system run-time, this test will measure the average time to process one image in the OR software. Like the OCR test, this test will be run on multiple sets of random images that are the same size, and all the resulting average times will also be averaged to get a more precise

measurement. The range for this test is anywhere between 5 to 20 milliseconds for each image. A result that is more than 10% longer than this range will be considered a failure.

Procedure:

1. Create a folder in Google Drive called “OR_Speed_Test” and inside it, place a single vehicle image changing the name to “test_1” and 5 folders with the following criteria:

   “test_2”: add 5 images

   “test_3”: add 10 images

   “test_4”: add 15 images

   “test_5”: add 20 images

   “test_6”: add 1000 images
2. Make sure that the “best.pt” file for the YOLO model is stored in content/drive/MyDrive/runs/detect/train/weights/best.pt
3. Run the cells in the following Google Colab Jupyter notebook and record the run-times for each step

2.3 Combined OR-OCR Run-time

Since the application aims to combine OR and OCR seamlessly, it is important to test the speed of the combined process for a single image to ensure the combined solution is efficient. This test will record the average total run-time for 10 trials of running the combined solution on a single image from a folder containing 20 images.

Procedure:

1. Run the application and select a folder containing vehicle images
2. Record the time it takes from selecting “choose” until the file(s) are displayed in the file navigation section
3. Select an image within the file navigation section
4. Record the time it takes from selecting the image until the output information is populated/displayed

3.1 Image Loading Speed Variance

Currently, the solution will be used as a tool for CampusParc data cleansing and not as a real-time LPR system which reads plates as they are scanned. It is estimated that the data cleansing would be required to work for either one image at a time or a folder with anywhere from 10-20 images. This test, which is closely related to the system run-time, will test the efficiency of the system by recording the time it takes to load files in the application for 1, 5, 10, 15, and 20 images. This is

not a pass/fail test, rather involves recording the variance in run-times for the different image quantities. Ideally, there should be little variance in the efficiency for the different image quantities.

Procedure:

1. Run the application and browse for a file
2. Select a single image file (jpg, jpeg, png)
3. Record the time it takes from selecting “choose” until the file(s) are displayed in the file navigation section
4. Repeat steps 2-3 but select a folder containing 5 images
5. Repeat steps 2-3 but select a folder containing 10 images
6. Repeat steps 2-3 but select a folder containing 15 images
7. Repeat steps 2-3 but select a folder containing 20 images

4.0 UI Set-up

The UI that the sponsors are going to receive has been delivered via GitHub. This is the final delivery, hence needs to test the display and analyze the performance on the front and back-end. For any software, the executable is tested using either demonstration or inspection.

Procedure:

1. Start with downloading the repository from GitHub that has a readme file with instructions to set up the GUI and required extensions.
2. Once the GUI and extensions are installed, open the GUI and start with selecting the folder in which the vehicle images are stored.
3. It should instantly show a scroll of jpeg, png, files to navigate.
4. Try the approval or correcting buttons to move to the next image automatically.
5. If you see a wrong read LP, correct it using the text box provided. It should update the backend dataset.
6. Change the folder location and repeat the test step 3 – 5

4.1 UI Feature Visibility

This UI test aims to validate that the vehicle information is displayed when an image is selected in the software application. The features include:

- Window title
- Browse button
- Image filenames in file navigation
- Original vehicle image
- License plate prediction
- Confidence score

- Plate correction option
- “Yes” and “No” buttons
- Cropped plate image

Procedure:

1. Run the application and ensure the window opens
2. The title “License Plate Recognition” is displayed at the top of the window
3. “Select a folder containing license plate images:” is displayed
4. A “browse” button is shown next to the selection prompt
5. Select “browse” and navigate to a folder containing vehicle images
6. The image files in the folder are displayed in the file navigation section
7. The original vehicle image and “Image selected:” and the file name are displayed
8. “Prediction:” and the prediction for the image are displayed
9. “Confidence:” is displayed and the value is a percentage
10. “Is this plate information correct?” is displayed
11. “Yes” and “No” buttons are visible below the plate information check

4.2 UI Navigation

In addition to the critical information, the confidence score, cropped plate image, and an option to manually correct the plate information are beneficial to include. This UI test will validate that when the program runs the original image, license plate prediction, confidence score, and a checkmark and “x” option are displayed. When the checkmark is pressed, the confidence score changes to 100%. If “x” is pressed, an input field shows up to allow the employee to enter the actual license plate and then press the checkmark to update the information. If the image is illegible, then when the “x” is pressed, and nothing is entered then the check is pressed an “invalid” response will be recorded. If all these steps pass, then the test passes.

Additional information:

- Ability to scroll through folder of images
- Ability to correct LP information manually
- Ability to view images within browsing folder before selecting
- Update the dataset and improve the prediction score if appeared again

Results

Following the test procedures described above, the following is the list of results collected by the team. Each test was run by an engineer and reviewed by others to validate the results.

Table 1.1 OR Accuracy with clear images

Test	Source	# images	Total Accuracy	Average Accuracy	Engineer
1	Flock	330			
2	Vaxtor	330			
3	ParkAssisst	330			

Table 1.2 OR accuracy with challenging images

Test	Source	# Images	Total Accuracy	Average Accuracy	Engineer
1	Flock	330			
2	Vaxtor	330			
3	ParkAssisst	330			

Table 1.3 OCR Accuracy

Test	Type	# Images	Total Accuracy	Average Accuracy	Engineer
1	Clear	200			
2	Night	200			
3	Blur	200			

Table 2.1 OCR System Run-time

Test	# Images	# "Clear" Images	#Accurate readings	Accuracy rate	Run-time per image (ms)	Total run-time (sec)	Engineer
1	1000	N/A	859	85.87%	33.602	33.602	SC & JL
1.1	200	193					JL

Table 2.2 OR System Run-time

Test	# Images	Total Run-time (ms)	Average Run-time per image (ms)	Engineer
1	1			
2	5			
3	10			
4	15			
5	20			
6	1000			

Table 2.3 Combined OR-OCR Run-time

Test	Total run-time (ms)
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
Average run-time	
Engineer	

Table 3.1 Image Quantity Variance

# Images	Actual loading time (ms)	Expected loading time (ms)	Variance	Engineer
1		N/A	N/A	
5		(actual for 1) * 5	(expected – actual)	
10		(actual for 1) * 10	(expected – actual)	
15		(actual for 1) * 15	(expected – actual)	
20		(actual for 1) * 20	(expected – actual)	

Table 4.0 UI Set-Up

Criteria	Pass/ Fail	Remark

One step download		
Set up the GUI		
Select a folder		
Navigate between folder paths		

Table 4.1 UI Feature Visibility

Feature	**Displayed ?**	**Remark**
“License Plate Recognition” window title		
“Select a folder containing license plate images:”		
“browse” button		
Images displayed in file navigation section		
original vehicle image		
“Image selected:” and file name		
“Prediction:” and prediction value		
“Confidence:” and value as percentage		
“Is this plate information correct?”		
"Yes" button		
Engineer		

Table 4.2 UI Performance

Criteria	Read	Remark
OR runtime		
OCR runtime		
GPU required		
Average confidence		