Loop Closure Detection Based on Semantic Templates

Jialin Li[1]

MSc Robotics and Computation

Supervisor: Professor Simon Julier, Ziwen Lu

Submission date: 26 Septermber 2022

Abstract

This paper investigates stability of semantic features for loop closure detection under different lighting conditions. The basis of loop closure detection is comparison of similarity between images. Current state-of-the-art SLAM systems, such ORB-SLAM2, measures image similarity based on low-level features such as pixel intensity, which is unstable under different lighting conditions. This paper developed an original image similarity measurement algorithm based on semantic templates. Experiments are carried out to compare performance of loop closure detection under different lighting conditions using semantic features against ORB features. A conclusion is drawn that semantic features have better stability under different lighting conditions than ORB features.

Contents

Chapter 1

Introduction

Current state-of-the art SLAM systems are keyframe-based, such as ORB-SLAM2 [9]. Keyframes are used to build local map and estimate camera pose. However, the map points built and camera pose estimations always have errors. As the system runs, these errors build up and the drift increases. Although BA(bundle adjustment) is frequently applied to optimize camera pose and 3D map points, it cannot completely cancel out the drifts. Current state-of-the art SLAM systems use loop closure to effectively eliminate drifts. Loop closure uses the current keyframe to detect if it's similar with any past keyframes stored in the system. If more than enough consecutive keyframes are found similar, SLAM system would realise that it's at a place it's been to before. In this case, loop closure is detected. Hence, the basis of loop closure detection is to find similarity between images accurately.

In real world, when the system travels back to the same place, lighting condition could be different from the last time it was there due to different time-of-the day, different weather condition or even different seasons. Most SLAM systems use low-level features such as pixel intensity to detect loop closure. Unfortunately, pixel intensity is sensitive to change in lighting condition. In order to detect loop closure under different lighting conditions, a high-level feature robust to change in lighting condition should be investigated.

Many studies have shown that semantic features are robust under different lighting condition. [12] [7] [11] Hence, in this paper, the stability of using semantic features for loop closure detection under different lighting conditions is investigated.

Main contributions of this paper

- developed an algorithm for single semantic template matching
- summarized criteria for semantic template matching
- developed an algorithm for image similarity measurement based on multiple semantic template matching
- verified that semantic features are more stable to lighting condition change than ORB features

The structure for the rest of this paper is as follows:

- *Chapter 2* reviews related literature on applying semantic features for loop closure detection
- *Chapter 3* describes a single semantic template matching algorithm. It also investigates properties of semantic template matching with single template matching problem.

- *Chapter 4* describes image similarity measurement algorithm based on multiple semantic template matching
- *Chapter5* describes experiment design to compare performance of loop closure detection under different lighting conditions using semantic features against ORB features
- *Chapter6* presents experiment results and their analyses
- *Chapter7* draws conclusion on this paper and gives suggestions for future work

Chapter 2

Related Work

Many studies have been carried out to tackle loop closure detection under changing lighting conditions with semantic information. Li *et al.* [5] used semantic segmentation to preprocess raw RGBD images to distinguish dynamic objects (such as cars, pedestrians) and static objects (such as buildings, traffic signs). Low dimensional semantic features and high dimensional semantic features are then extracted from those preprocessed images with CNN (convolutional neural network). Candidate key-frames are filtered first by comparing their high-dimensional features with the high-dimensional features of the current frame. The key-frames left are then compared with current frame by their low-dimensional features. The key-frames left are candidates for loop closure. Islam *et al.* [3] used CNN to extract intereted regions from semantic images. ORB features are then extracted from the interested areas and BoW vectors are constructed from those ORB features. Then image similarity is computed the same way as in ORB-SLAM2. Merrill *et al.*[8] developed a multi-decode VAE based network which is trained to extract a specially designed image descriptors. These image descriptors are composed of local features encoding visual appearance and semantic information. K-nearest neighbor search is then performed to detect loop closure. Zhu *et al.* [13] used Lite-shuffleNet network to extract semantic feature vectors. The similarity between two semantic feature vectors are computed with cosine method. $CoSim(m, n) = \frac{m.n}{||m||||n||}$, where m, n are two feature vectors for comparison, $CoSim$ is similarity between m and n. The performance is then compared with semantic features extracted with VGG-16 network.

While good results are achieved with these methods, they all reply on the CNN network to extract semantic features from images. The performance of CNN network heavily depends on the network structure and how the network is trained. To achieve high accuracy, the network needs to be trained on specific scenarios. However, if application environment changes, the network would need to be retrained, otherwise the performance could be unstable. This paper intend to develop a method that could extract semantic features directly from semantic images without the need to pre-train a network. Semantic segmentation tend to be unstable near object boundaries due to smoothing. Hence, rather than feature points, image templates (patches of an image) are extracted which averages out small errors at object boundaries. In this way, similarity between two images is measured by the number of templates matched from those images.

Chapter 3

Investigation of Semantic Template Matching Properties

In order to investigate properties of semantic template matching, we start with a single template matching problem. The dataset used is sequence10 of kitti360 dataset [4] which features images captured in urban environment.

Given a template extracted from a reference image, template matching is to find a patch from a query image that matches best with it regardless of orientation and scale change. PTAM uses a warping matrix based on camera pose to warp the image patch to the same size and orientation as the template before matching them. In this investigation step, the image adjacent to the reference image in the consecutive image sequence is chosen as the query image so we can assume it has similar scale and orientation as the reference image.

3.1 Semantic Template Matching Algorithm

Normal algorithms for template matching based on pixel intensity, such as MAD, SAD, SSD are not suitable for template matching based on semantic features. The reason is difference in pixel intensity just represents difference in magnitude while difference in semantic value represents different objects. For instance, '7' represents 'sky' while '8' represents 'road'. Hence, algorithms suitable for semantic template matching need to be found at first.

3.1.1 Conditional Probability

Given a template and an image patch of the same size, each pixel in the image patch matches with the pixel at the same position in the template, forming a pixel pair. In template matching, template is given. Hence, the probability that a pixel (p) in the image patch matches with a pixel (t) in the template is conditional probability, $P(p|t)$. The likelihood ($l(P|T)$) of an image patch (P) being the match of the template image (T) is measured as the sum of conditional probability of all pixel pairs in the image patch and the template image. Note this algorithm assumes the conditional probability of each pixel pair is independent from all the other pixel pairs.

$$l(P|T) = \sum_{p_i \in P, t_i \in T} log(P(p_i|t_i)) \tag{3.1}$$

Conditional probability of each pixel pair depends on the specific value in both pixels. For ex-

Table 3.1: Example of histogram matrix

	tree	pole	sky	building	wall	cloud
tree	500	100	3	5	6	7
pole	100	700	5	6	8	7
sky	10	2	600	20	5	100
building	20	13	2	600	150	20
wall	3	21	2	150	500	20
cloud	15	4	150	20	3	600

ample, the pixel in the image patch has a value that represents 'tree', $p_i = tree$. The corresponding pixel in the template has a value that represents 'sky', $t_i = sky$. The conditional probability that $p_i = tree$ matches with $t_i = sky$ is different from the conditional probability that $p_i = cloud$ matches with $t_i = sky$. In order to get conditional probability distribution for every combination of semantic labels, a histogram matrix is build from all images in sequence10 of kitti360 dataset.

3.1.2 Histogram Matrix Construction

kitti360 dataset provides 3D pointcloud model of sequence10, camera intrinsic matrix and ground truth camera pose for each image in the sequence. For each image, points from point clouds are projected to that image according to corresponding camera pose. Points are correlated with pixels they project onto. The same points are then projected to the next image in the sequence. A pixel in the first image and a pixel in the second image are paired up if they correlates to the same 3D point. The pixel pairs are then grouped by their pixel value combination. The number of pixel pairs in each kind of combination are added to the corresponding cell in histogram matrix. For example, 1 represents "tree", 3 represents "sky". If there are 100 pixels pairs with the first pixel being 1 and the second pixel being 3, 100 is added to the cell at row 1 and column 3 in the histogram matrix. An example histogram matrix is shown in *Table 3.1.*

Conditional probability of a specific semantic label combination is calculated as the corresponding cell in the histogram matrix divided by the sum of the row. For example, the conditional probability of 3 ("sky") matches with 1 ("tree") given 1 ("tree") is the cell at row 1 column 3 divided by the sum of all cells in row 1. In the example histogram matrix, it will be $0.00483 = 3/(500 + 100 + 3 + 5 + 6 + 7)$. The sum of row 1 is the number of times "tree" appears in the image sequence. Each cell in row 1 is the number of times the corresponding semantic label matches with "tree". Hence, the conditional probability of j matches with i given i, is the cell at row i column j divided by the sum of row i.

The histogram matrix is not built on one specific image but all the images in the sequence. Not only it generates conditional probability of each kind of semantic label combination, but also reflects the performance of the classifier network. For example, in the example histogram matrix (*Table 3.1*), the conditional probability of "tree" matches with "tree" given "tree" is higher than the conditional probability of any other semantic labels matches with "tree" given "tree". This means the classifier network tends to classify "tree" as "tree" more than any other semantic labels. It can also be observed the conditional probability of "pole" matches with "tree" given "tree" is higher than any other semantic labels except "tree". It shows the classifier network found some connections between "pole" and "tree", so it sometimes classify "tree" as "pole". It also found some connections between "building" and "wall", and "sky" and "cloud". The performance of the classifier network may not be very accurate but the histogram matrix reflects the classifier network performance accurately.

3.2 Single Template Matching

3.2.1 Template Extraction

A template is searched in the reference image with sliding window method. The sliding window slides from left to right on each row, from top row to bottom row across the reference image. The search stops once a valid template is found. At the beginning, a valid template is a template with more than 1 type of semantic label in it. Template shape is rectangle because it's easy to analyse. Template size is chosen arbitrarily at first. The sliding window is in the same size as the template such that the region within the sliding window is a template.

3.2.2 Template Match

The match of the template extracted from a reference image is searched in a query image. Sliding window method is used to search for the template match because it's straight forward and easy to analyse. Sliding window slides across the entire query image, generating a matching score at each position and store it in a matrix. The matching score is calculated with the algorithm in *Section 3.1.* The size of the matrix is $(W - w + 1) \times (H - h + 1)$, where W and H are the width and height of the query image, w and h are the width and height of the template. By transforming this matrix into a heat-map, the matching score distribution can be visualised. This way, it's clear to see in which region the template has good match and in which region it has bad match.

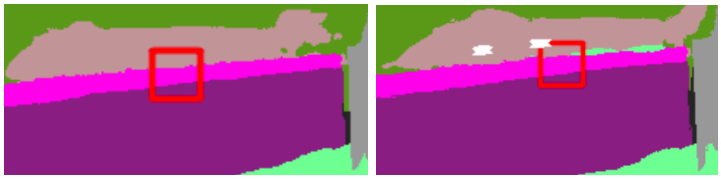
3.2.3 Template Matching Properties and Criteria

Based on experiments with single template matching, the following properties are found:

- templates with less than 3 types of semantic labels cannot be matched precisely. Usually there are many regions with high matching score for such templates
- even for templates with 3 or more types of semantic labels, if they follow a certain pattern, they cannot be matched precisely. Specific description of the pattern can be found in *Section 3.2.4*
- templates with dynamic objects such as cars, cannot be matched accurately because dynamic objects move across frames
- templates with semantic labels with low confidence level cannot be matched accurately. By confidence level, it means the conditional probability of a semantic label being matched to itself, e.g. $P(tree|tree)$
- templates with semantic labels with very high confidence level cannot be matched accurately. The reason is those semantic labels are too dominant, such that the algorithm only focus on matching those semantic labels correctly while the other semantic labels are ignored
- templates with large size usually cannot be matched accurately. The reason is a template in reference image may not have an exact copy in query image due to view point change. An example is shown in *Figure 3.1. Figure 3.1 (a)* shows a template surrounded by a red box. The middle of the template is a traffic sign, the top half is sky, and the bottom half is vegetation. *Figure 3.1 (b)* shows the same view as *Figure 3.1 (a).* However, we cannot find a template with the same distribution as the template in *Figure 3.1 (a)* anymore. The larger the template is, the more difficult to find a match that's similar to it

(a) template (b) match result

Figure 3.1: Large template usually unstable



(a) template (b) match result

Figure 3.2: Indistinguishable template pattern

Based on those properties, the following criteria are built for solid single template matching:

- templates must have at least 3 different types of semantic labels. Each type of label must occupy at least 20 % of template to avoid outliers. The threshold 20 % is chosen empirically
- templates must not follow the pattern described in *Section 3.2.4*
- templates must not contain dynamic objects
- templates must not contain semantic labels whose confidence level is lower than 0.6. This confidence level threshold is chosen empirically
- the image patch that matches of the template must have the same types of semantic labels as the template, while each type of label occupying at least 20 % of the image patch
- a small template is in favor. A suitable template size is empirically found to be 15×15

3.2.4 Indistinguishable Patterns

Figure 3.2 (a) shows a template surrounded by a red box. It can be seen the template contains 3 types of semantic labels but their distributions are almost parallel with each other. Such template cannot be matched precisely as shown in *Figure 3.2 (b)*, where the white region is composed of many white crosses. Each white cross marks a position of the left-up corner of an image patch with a matching score over 95 % of the maximum matching score.

A filter algorithm is applied to identify this kind of pattern. If a template's one pair of opposite edges (left and right, or up and down) have more than 1 type of semantic labels in common, while the other pair of opposite edges have only one type of semantic label each, this template would be filtered out. This algorithm is able to filter out most types of patterns that are indistinguishable due to lack of interaction among different semantic labels.

Chapter 4

Image Similarity Measure with Semantic Templates

Similarity between two images is measured by the number of templates matched. Instead of extracting templates from a reference image and try to find their matches in a query image, our method extracts templates from both reference image and query image, and matches them directly with each other.

Similarity between two images is the sum of templates matched in both images divided by sum of all templates extracted from both images.

$$S = \frac{M_{1\to 2} + M_{2\to 1}}{T_1 + T_2} \tag{4.1}$$

where S is the similarity score, $M_{i\to j}$ is the number of templates in image i matched in image j, T_i is the number of templates extracted from image i. Since templates are matched one-to-one, the number of templates matched in both reference image and query image are the same, $M_{1\to 2} = M_{2\to 1}$.

4.1 Multiple Semantic Template Extraction

The following is the algorithm of multiple template extraction:

- templates are selected from an image by sliding window method. The sliding window slides from left to right on each row, from top row to bottom row across the entire image
- an algorithm is designed to make sure the selected templates do not overlap with each other
- templates satisfying the criteria in *Section 3.2.3* are very rare. Usually less than 5 per image. Hence, templates that have only 2 types of semantic labels are also extracted. For such templates, each type of label must occupy at least 40 % of the template. Other than the first two items, this type of template must satisfy the rest of the criteria in *Section 3.2.3*
- templates satisfying all the criteria in *Section 3.2.3* are called "fine templates". Templates with only 2 types of semantic labels are called "coarse templates".
- templates are stored in the order they are selected, e.g. the first template selected is stored at the beginning. Coarse templates and fine templates are stored separately

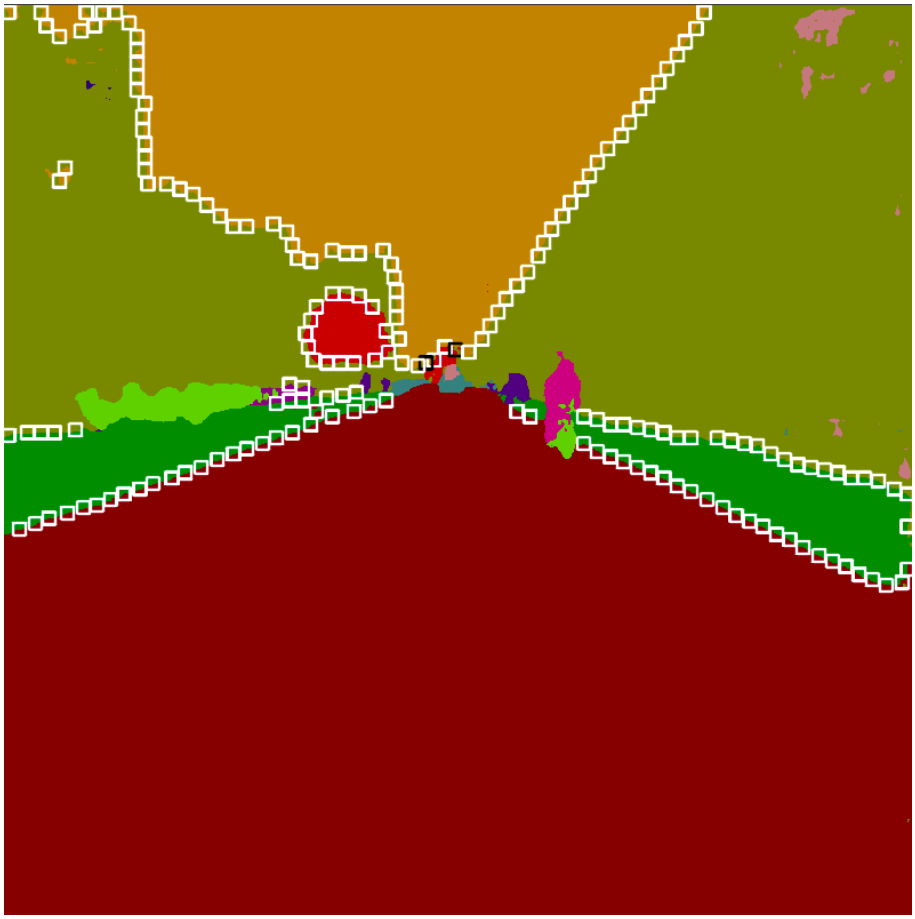


Figure 4.1: Multiple template extraction in summer

- for each template, only the position of the left up corner of the template in the image, the types of semantic labels in the template and a Boolean number are stored. The template storage structure for coarse templates and fine templates are shown in *Equation 4.2*. x_i and y_i are coordinate of template i in image, s_i^j is a type of semantic label in template i, b_i is a Boolean number indicating if template i has been matched yet. Note that fine template may have more than 3 types of semantic labels. This type of feature vector stores both semantic and geometric information of the templates with a few digits instead of an entire image patch.

$$\begin{aligned} coarse\ \ template: &\quad \{[x_1, y_1, s_1^1, s_1^2, b_1], [x_2, y_2, s_2^1, s_2^2, b_2], [x_3, y_3, s_3^1, s_3^2, b_3], ...\} \\ fine\ \ template: &\quad \{[x_1, y_1, b_1, s_1^1, s_1^2, s_1^3, ...], [x_2, y_2, b_2, s_2^1, s_2^2, s_2^3, ...], ...\} \end{aligned} \tag{4.2}$$

- *Figure 4.1* shows templates extracted from a semantic image for demonstration of the multiple template extraction algorithm. White boxes are coarse templates, and black boxes are fine templates. Note that fine templates and coarse templates could overlap with each other since they are stored separately and matched separately.

4.2 Multiple Semantic Template Match

Semantic images capturing the same view are similar under different lighting conditions. *Figure 4.2* and *Figure 4.1* show semantic images of the same view captured in winter and in summer

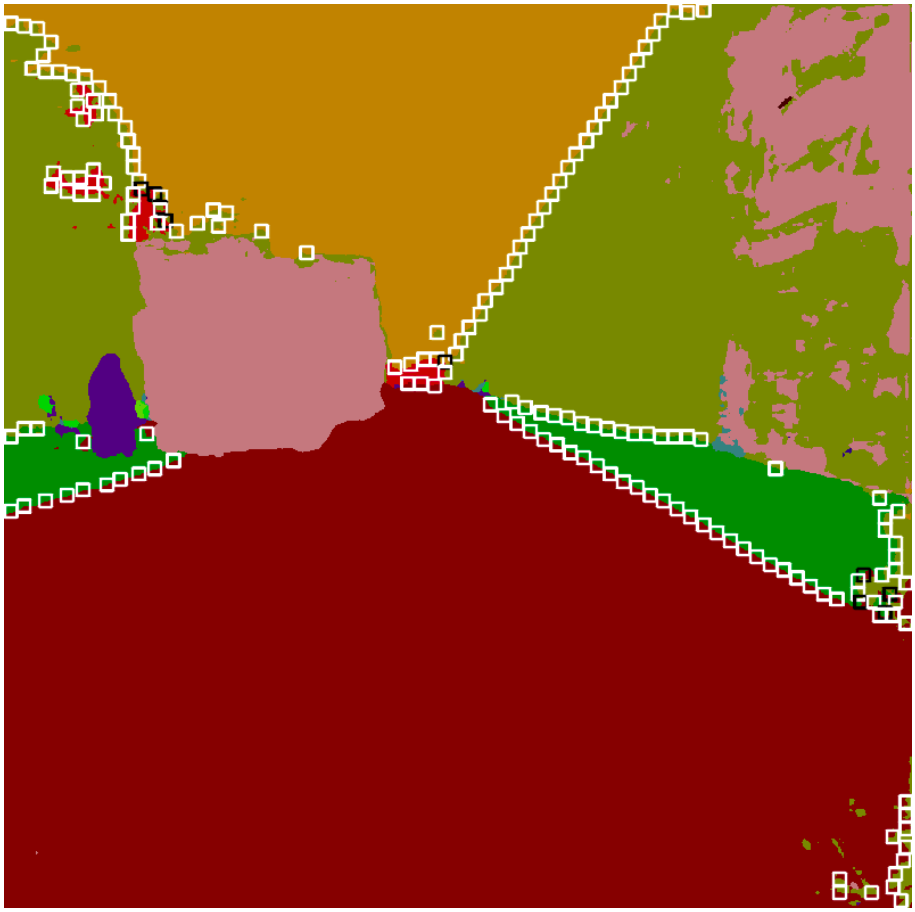


Figure 4.2: Multiple template extraction in winter

respectively. Although there are some differences, it can still be visually recognised that they are of the same view. However, when lighting condition change is too extreme, semantic segmentation may be unstable since it is based on pixel intensities. *Figure 4.3* shows semantic image of the same view as *Figure 4.1* and *Figure 4.2* except it's captured at night. Due to the change in lighting condition is too extreme, it's very hard to recognise that *Figure 4.3* captures the same view as *Figure 4.1*.

Although semantic segmentation stability depends on the quality of the classifier network, it cannot be expected that the semantic segmentation network always performs ideally. Hence, template matching must have robustness against variation in semantic images due to change in lighting condition.

Although *Figure 4.1* looks very different from *Figure 4.3*, there are still some regions that have similar appearance, as shown in *Figure 4.4*. Our algorithm tends to find out as many similar regions as possible in the form of templates matched in both images. Detailed multiple semantic template matching algorithm is described in the following subsections.

4.2.1 Template searching order

There are two sets of templates, one set is extracted from reference image (T_r), the other set is extracted from query image (T_q). Templates in T_r are matched one by one, from the beginning of the template list to the end. For each template in T_r, its match is searched in T_q from the beginning to the end.

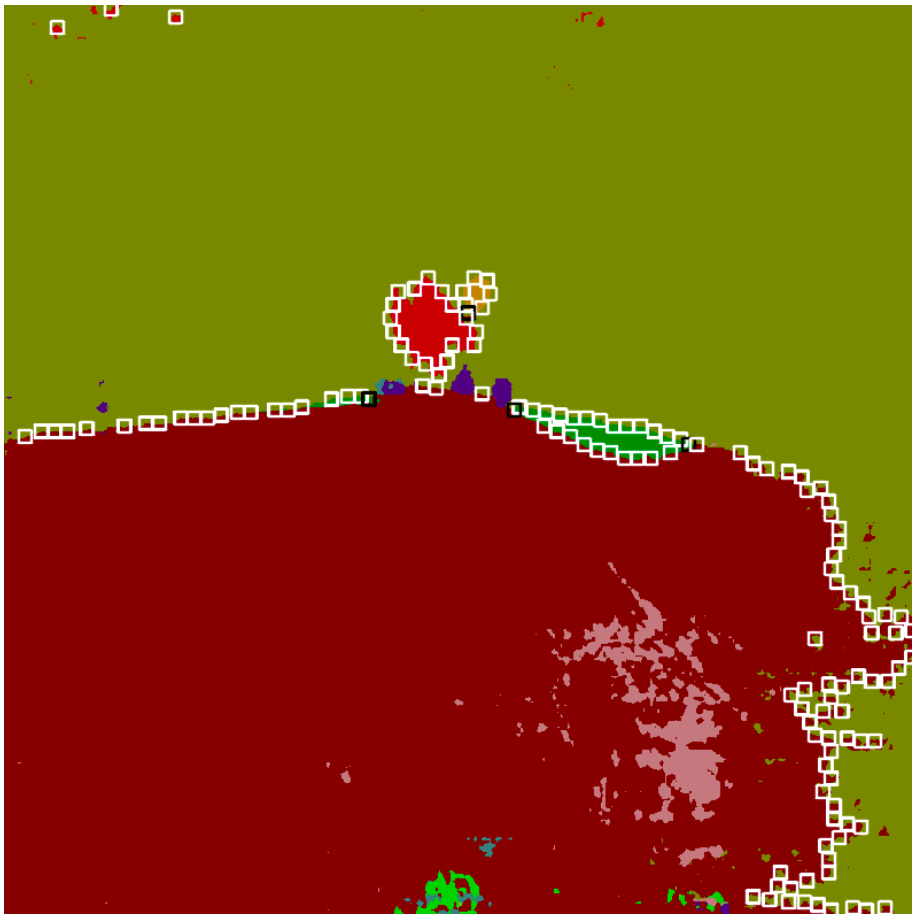


Figure 4.3: Multiple template extraction at night

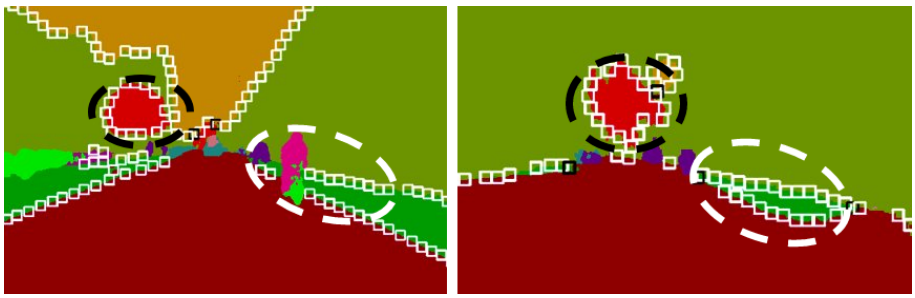


Figure 4.4: Similar regions in summer image and night image

This searching order is not chosen arbitrarily. Templates are stored in the same order as they are extracted. Templates from both images are extracted in sliding window method. This means the template at the start of T_r is likely to find a match near the start of T_q. Once a template from T_q matches with a template from T_r, both templates are labeled as matched. The next template from T_r still searches for its match from the start of T_q. The templates in T_q that are already matched will be skipped. In addition, if there are multiple templates in T_q that could be matched with the current template in T_r, this searching order ensures the template in T_q close to the current template in T_r would be matched first. Hence, this searching order provides a fast and accurate way of multiple template matching.

4.2.2 Fine template matching algorithm

According to the second last item in template matching criteria from *Section 3.2.3*, two templates must have the same types of semantic labels to be matched together. Fine templates have at least 3 types of semantic labels. Given the small template size, this sole matching criteria is able to match fine templates accurately.

There is no need to transform or warp templates before matching. The reason is the matching criteria of fine template only focuses on what types of semantic labels are contained in the template. Hence, this matching criteria is already robust to scale and orientation change. Transformation or warping of the template makes no difference on the matching result. The second reason is transformation and warping are normally carried out according camera pose transformation between the reference image and the query image. However, by doing loop closure, we are trying to correct camera pose estimation. Hence, it doesn't make sense to correct camera pose based on camera pose.

4.2.3 Coarse template matching algorithm

Coarse template is not as identical as fine template since it has only two types of semantic labels. Other than the requirement that two templates must be of the same type to be matched, an extra matching criteria is developed to match coarse templates accurately and efficiently. The templates that have the same types of semantic labels are of the same type.

By inspecting the template distribution in similarity regions in *Figure 4.4*, other than the fact that they are the same type of templates, another clear clue is they have similar geometric distribution. Hence, the extra matching criteria exploits geometric information among coarse templates.

Rather than gathering the geometric information of all templates within a region, we focus on geometric correlation among only a few templates that are close together and belong to the same type (have the same types of semantic labels). The advantage of focusing on small regions is it's more robust and stable than large regions, especially when the change lighting condition is big. Even if some small regions become unstable in semantic segmentation, there are still many other small regions available. For example, in *Figure 4.4*, although the regions bounded by the white circle has similar geometry distribution, during real template matching, the region in the left image wouldn't be defined by the coarse templates within the white circle. Instead, all the coarse templates surrounding the green region would be grouped together, as shown in *Figure 4.5*. This time, the geometry of those two regions doesn't look similar as each other anymore.

In this algorithm, a small region is defined by three templates of the same type that are close together. Coarse templates and fine templates are stored separately. The coarse template list of

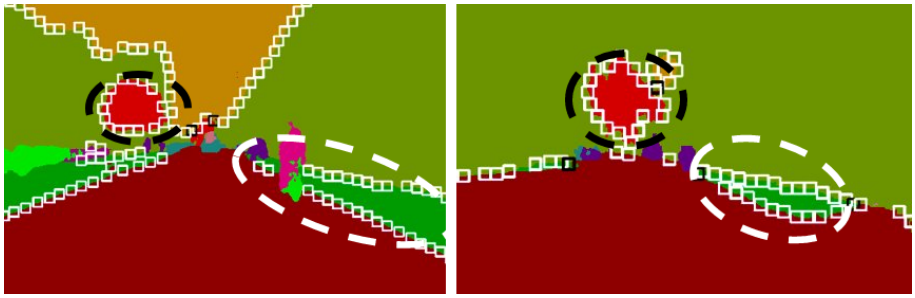


Figure 4.5: Large grouped regions in summer image and night image

the reference image is represented as T_r. The coarse template list of the query image is represented as T_q. For a coarse template from T_r that is to be matched and represented by r_1, the template nearest to r_1 is represented by r_2, and the template the second nearest to r_1 is represented by r_3. These three templates form a region, R_r. These three templates are of the same type. From T_q, the first template that haven't been matched and is of the same type as r_1, is represented as q_1. The template nearest to q_1 is represented as q_2, the template the second nearest to q_1 is represented as q_3. These three templates form a region, R_q. Again, they are of the same type.

The three templates within a region are then connected by three lines. If the ratio of the length of these three lines are similar between R_q and R_r, then R_q and R_r are considered as similar regions. r_1, r_2, r_3, q_1, q_2, q_3 are all labeled as matched. Note that templates that have been matched cannot be matched again, but they can be used to form a region. In straightforward explanation, r_1 and q_1 must be unmatched at the beginning. r_2 and r_3 could be matched or unmatched. If r_2 is matched, q_2 must also be matched, if r_2 is unmatched, q_3 must also be unmatched. The same for r_3 and q_3. In the end, the number of templates matched are counted by the number of templates labeled as "matched".

Figure 4.6 shows the coarse template match of two small regions. The matched templates are shown in red boxes. It can be seen the small regions formed by 3 coarse templates are matched accurately. *Figure 4.7* shows the coarse template match of the entire image. It can be seen that most similar regions are recognised.

Just like fine template matching algorithm, coarse template matching algorithm is also robust to scale and orientation change since it depends on the ratio of distance among templates.

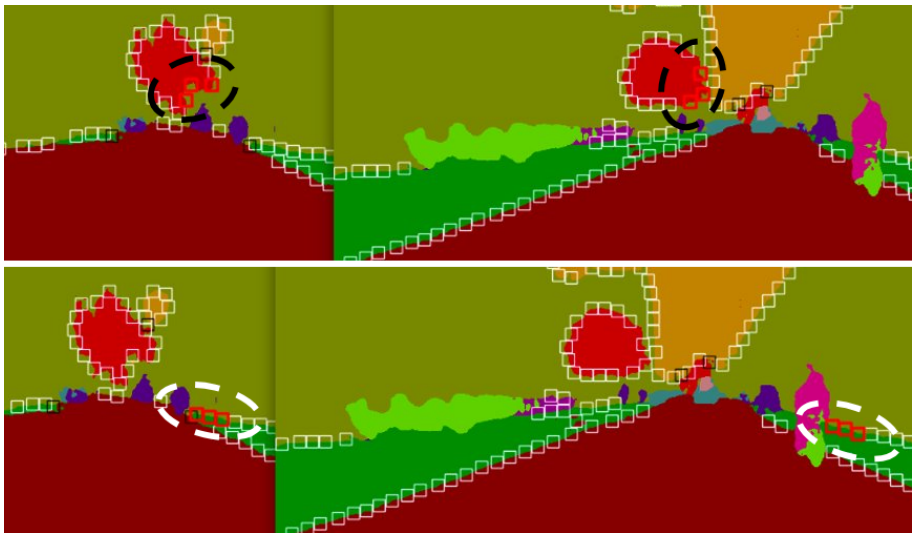


Figure 4.6: Coarse template match of some regions

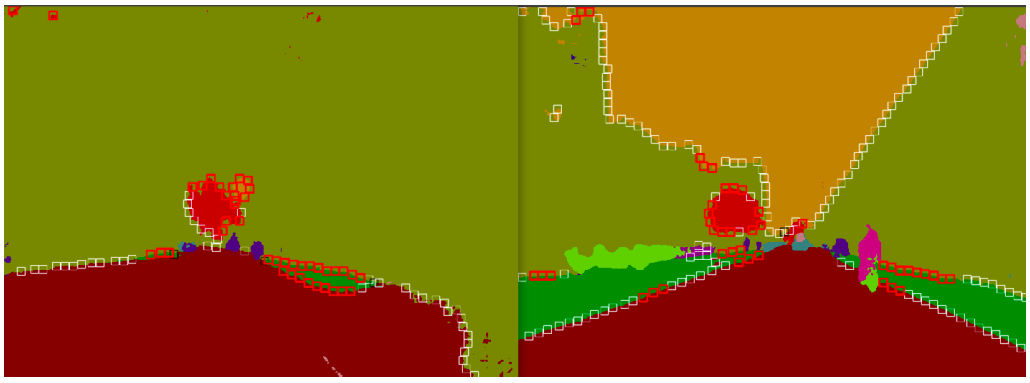


Figure 4.7: Coarse template match of the entire image

Chapter 5

Algorithm Evaluation

To evaluate the performance of the image similarity measure algorithm described above, it's tested in loop closure detection against image similarity measure algorithm in ORB-SLAM2.

5.1 Dataset Description

The dataset used is "RobotCar-Seasons Dataset" [10] which is a subset of "Oxford RobotCar Dataset" [6]. The images are captured by cameras mounted on a car driving through Oxford for over 100 repetitions over a period of over a year. "RobotCar-Seasons Dataset" consists of images taken in 9 traversals of the same 8707m long path. Each traversal is taken under a different lighting condition: dawn, dust, night, night-rain, overcast-summer, overcast-winter, rain, snow, sun. The path is divided into 49 non-overlapping sub-sections. In each sub-section, a reference position is picked and all images from the 9 traversals with INS poses within 10m of that position are gathered in a text file. Images gathered around the 49 picked positions in each of the 9 traversals are stored separately. The structure of the dataset is shown in *Figure 5.1*. The images in the dataset are converted to semantic images with Dkaka Deeplab network[2], and the model is pre-trained from CityScape dataset [1].

5.2 Experiment Design

Two experiments are carried out. *Experiment1* compares loop closure detection performance between ORB features and semantic features under **the same** lighting condition. It is carried out with each kind of lighting condition listed in the dataset. *Experiment2* compares loop closure detection performance between ORB features and semantic features under **different** lighting conditions. There are 9 lighting conditions in total, each one is compared with 8 other lighting conditions in *Experiment2*.

For semantic features, experiments are carried out with semantic images, and image similarity is calculated with the algorithm mentioned in *Chapter 4*. For ORB features, experiments are carried out with the original images from the dataset. Image similarity is calculated with BoW (bag of words), which is the method used in ORB-SLAM2. In order to compute BoW in the most similar way as ORB-SLAM2, codes are extracted directly from ORB-SLAM2 to extract ORB features and compute BoW. The vocabulary dictionary used to compute BoW is built from all images from the "RobotCar-Seasons Dataset".

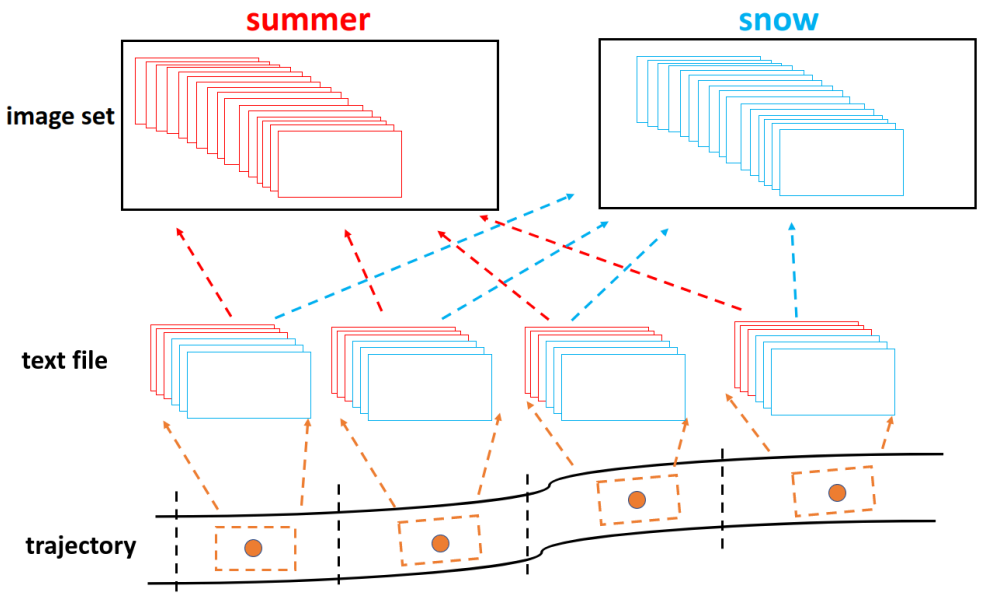


Figure 5.1: Dataset structure

5.2.1 Loop Closure Detection Algorithm

Algorithm of loop closure detection is the same for both experiments. For each lighting condition to be tested, which is referred as reference condition, a path sub-section is selected as the reference section. The last image in the reference section under the reference condition is the image used to detect loop closure. This image is referred as current frame, $cFrame$. A similarity score is calculated with between $cFrame$ and each other images in the reference section under reference condition. The minimum similarity score is recorded as $minScore$ and it's used as threshold to determine if another image is similar enough with $cFrame$. Images under another lighting condition is then used as query images to detect loop closure with $cFrame$. In *Experiment1*, the query condition is the same as the reference condition. In *Experiment2*, the query condition is different from the reference condition. The query images are grouped by the path sub-sections they belong to. $cFrame$ is then compared with each query image. If the similarity score between $cFrame$ and a query image is higher than $minScore$, the query image is similar enough with $cFrame$ and the path sub-section this query image belong to is added to a primary candidate list for further loop closure detection. For each path sub-section in the primary candidate list, the similarity score between $cFrame$ and each image in that sub-section is added up as a total score for this path sub-section. The highest sub-section score is recorded as $maxScore$. If the score of any path sub-section is higher than 75% of the $maxScore$, this path sub-section is detected as a loop closure candidate. A graph demonstration of the algorithm is shown in *Figure 5.2*. The pseudo-code of the algorithm is shown in *Algorithm 1*.

The performance of loop closure detection is defined by recall rate and accuracy. Recall rate is the number of times the reference path sub-section is detected as loop closure candidate over the number of tests carried out. Accuracy is the number of reference path sub-section detected as loop closure candidate over the total number of sub-sections detected as loop closure candidates. Experiment with each reference condition is carried out multiple times with different path sub-sections as reference section. The performance of loop closure detection for this reference condition is the overall performance of all the tests carried out with this condition.

Algorithm 1: Loop Closure Detection

Input: $refC$: reference condition
$refS$: reference path sub-section
$cFrame$: the last image in $refS$ under $refC$ is used as current frame to detect loop closure
$conFrames$: all the other images in $refS$ under $refC$
$qryImg$: images under query condition, grouped by the path sub-section they belong to

Output: $loopCandidates$: the path sub-sections that are detected as loop closure candidates

$minScore = minSimilarity(cFrame, conFrames)$ // $minScore$ is minimum similarity between $cFrame$ and $conFrames$
$primaryCandidates = [\ \ ]$ // list of candidates for further detection
for $subsection \in qryImg$ **do**
  for $img \in subsection$ **do**
    $score = compare(cFrame, img)$ // $compare(frame1, frame2)$ calculates similarity score between $frame1$ and $frame2$
    if $score > minScore$ **then**
      $primaryCandidates.add(subsection)$
    end
  end
end
$maxScore = getMaxScore(primaryCandidates)$ // calculate the score for each subsection in $primaryCandidates$ and record the maximum score as $maxScore$
$loopCandidate = [\ \ ]$ // the subsections detected as candidates for loop closure
for $subsection \in primaryCandidates$ **do**
  if $score\ of\ subsection > maxScore \times 0.75$ **then**
    $loopCandidates.add(subsection)$
  end
end
return $loopCandidates$

5.2.2 Comparison with Loop Closure Detection Algorithm in ORB-SLAM2

The loop closure detection algorithm in this paper is a mimic of the loop closure detection algorithm in ORB-SLAM2. The main differences between them and their potential effects on the performance of loop closure detection are listed below.

- Before loop closure detection, ORB-SLAM2 filters out candidate images from all the images stored. In our algorithm, since the number of images are not very massive, all the images are selected as candidate images
- In ORB-SLAM2, $minScore$ is calculated between $cFrame$ and images that are connected to it (share more than 15 map points). In our method, $minScore$ is calculated between $cFrame$ and all the other images in the same sub-section. According to the dataset description, images in the same sub-section are all within 10m of a chosen position. Hence, these images can be assumed to be connected to $cFrame$.
- ORB-SLAM2 chooses primary candidates by $minScore$ and another criteria called "common words". Our algorithm only used $minScore$ to select primary candidates. It might reduce accuracy since the standard of primary candidates is lowered, but it will not affect recall rate.

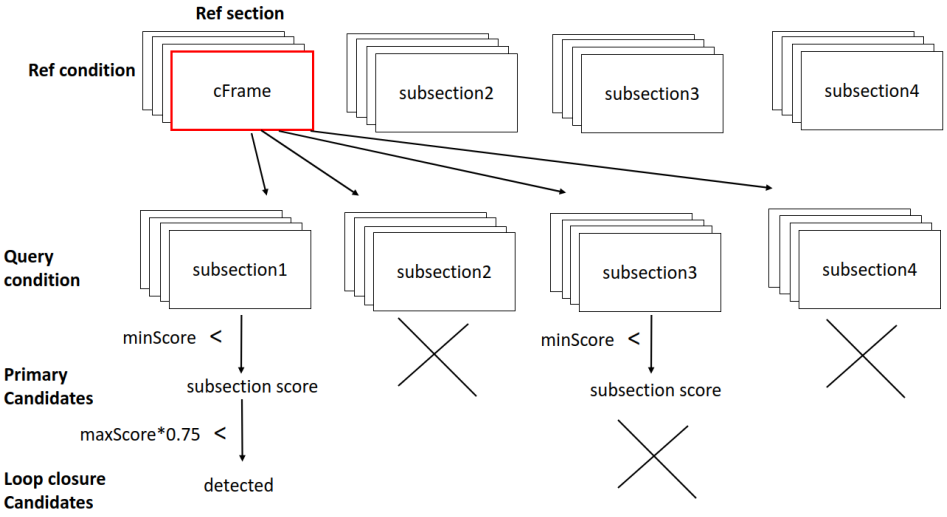


Figure 5.2: Loop Closure Detection

- For every image in the primary candidate list, ORB-SLAM2 finds 10 images that are most connected to it (share the most map points) and put them in a group. In our algorithm, the images in primary candidate list are already grouped by the sub-section they belong to. As mentioned before, images in the same sub-section can be assumed to be connected.

Chapter 6

Experiment Results and Discussion

6.1 Experiment1 result analysis

Figure 6.1 shows the accuracy and recall rate of loop closure detection under the same lighting condition for all lighting conditions in the dataset.

6.1.1 Accuracy analysis

It can be observed the accuracy with ORB features is 100% under all lighting conditions which shows ORB feature. The accuracy with semantic features are lower than 50% for all lighting conditions. A conclusion can be easily drawn that loop closure under the same lighting conditions has higher accuracy with ORB features than with semantic features.

For semantic features, it can also be observed the accuracy at "night-rain" condition is the lowest, and the accuracy at "night" is the second lowest. Recall that loop closure candidates are selected by a threshold $minScore$, which is the minimum similarity between the image for loop closure detection and all the other images in the same path sub-section under the same lighting condition. As shown in *Figure 4.3*, semantic images at night have much less identifiable features than semantic images at good lighting conditions. Hence, semantic images at different path sub-sections at night are not very identifiable from each other. Lots of images would have similarity score higher than the $minScore$, which means more images are detected as loop closure candidates and the accuracy level is low.

6.1.2 Recall rate analysis

For both semantic and ORB features, the recall rate under all lighting conditions are very high. The recall rate at "night-rain" condition is the only one that's not 100% for both semantic and ORB features. Compared with other condition, "night-rain" is the most challenging condition. As mentioned before, images at night have much less identifiable features than images captured in good lighting condition. In a night after rain, water on the ground could raise glares and form reflection of other objects on the street.

From the overall performance in *Experiment 1*, a conclusion can be drawn that ORB features are more suitable than semantic features for loop closure detection under the same lighting condition.

6.2 Experiment2 result analysis

As mentioned before, there are 9 lighting conditions in total, each one is compared with 8 other lighting conditions in *Experiment2*. Hence, for each lighting condition, loop closure detection is carried out 8 times with different lighting condition in each time. Accuracy and recall rate are calculated and recorded each time. Hence, each lighting condition has 8 sets of accuracy and recall rate for each different lighting condition and an overall accuracy and recall rate. *Figure 6.2* shows the overall accuracy and recall rate for each lighting condition. The 8 sets of individual accuracy and recall rate for each lighting condition are shown in *Figure 6.3* to *Figure 6.11*.

6.2.1 Accuracy analysis

From *Figure 6.2(a)*, it can be seen that for both semantic and ORB features, the accuracy at all lighting conditions are much lower than the accuracy at corresponding lighting conditions in *Experiment1*. This is an obvious evidence that change in lighting condition affects the performance of loop closure detection.

The overall accuracy and accuracy for each individual lighting condition is higher with semantic features than ORB features which is opposite to the performance in *Experiment1*. This means under different lighting conditions, semantic features are more stable than ORB features.

The overall accuracy of both semantic features and ORB features are the lowest at "night" and "night-rain". The reason is, as explained above, less identical features could be extracted from images captured at night. Hence, the *minScore* is very low which causes more images detected as loop closure candidates and accuracy drops.

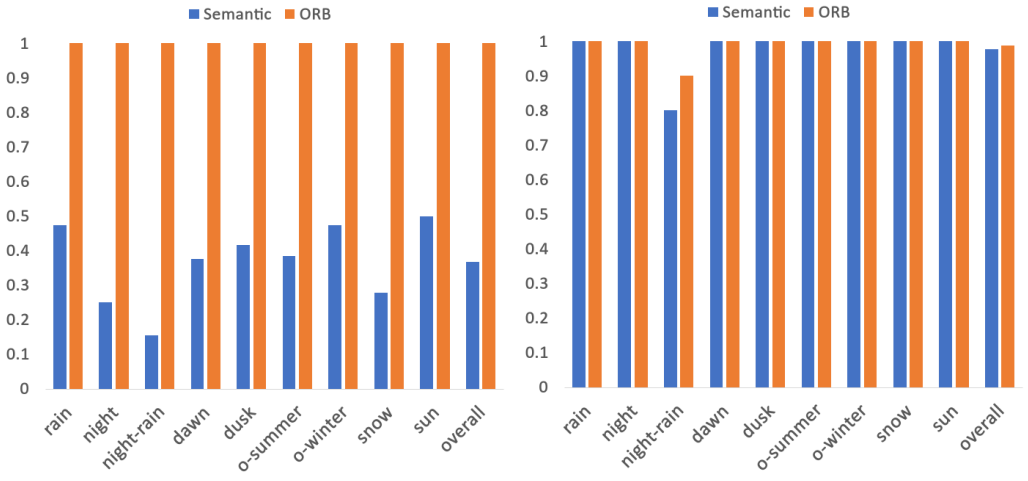
6.2.2 Recall rate analysis

From *Figure 6.2(b)*, it can be seen the recall rate for both semantic features and ORB features are lower compared to *Experiment1*. Again, this shows the performance of loop closure detection is affected by change in lighting condition.

Under "rain", "night", "overcast-summer", "overcast-winter" and "snow", the recall rate is higher with ORB features than with semantic features. Under "night-rain", "dawn", "dusk", and "sun", the recall rate is higher with semantic features than with ORB features. However, by inspecting the individual recall rate for each lighting condition except "night" and "night-rain" (*Figure 6.3*, *Figure 6.6* to *Figure 6.11*), it can be seen that other than condition "night" and "night-rain", the recall rate with semantic features are higher than ORB features for most of the time. Hence, other than "night" and "night-rain", the recall rate for loop closure detection under different lighting conditions is higher with semantic features than ORB features.

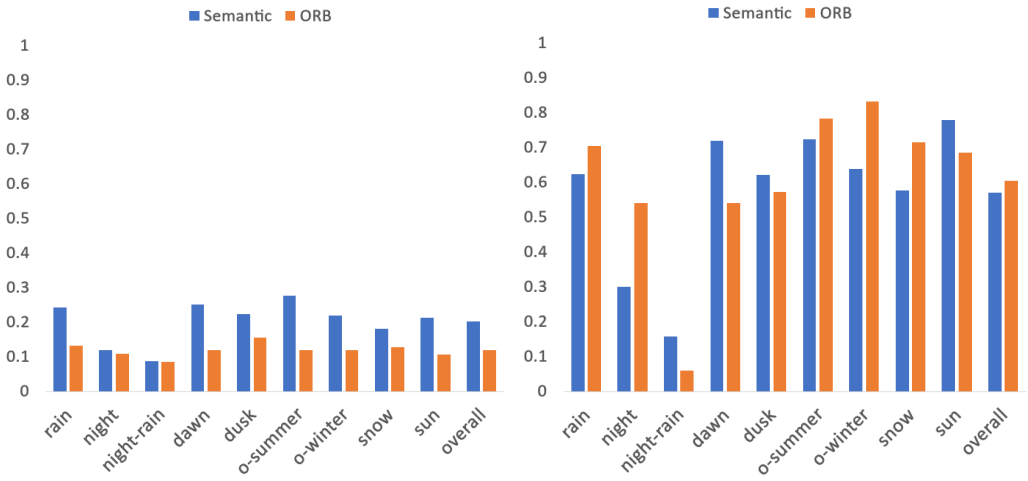
The reason why recall rate is usually low with semantic features at "night" and "night-rain" is that the semantic images at "night" and "night-rain" have very poor quality, see *Figure 4.3*, *Figure 4.2*, and *Figure 4.1* for example. Although our method manages to extract as many similarity regions as possible, the features within the semantic image are too few and too poor. Hence, even for images in the same path sub-section, the similarity between a semantic image captured at night and a semantic image captured in other lighting conditions are still very low.

A different recall rate trend can be found in *Figure 6.4* and *Figure 6.5*. In *Figure 6.5*, the reference condition is "night", the recall rate for "night-rain" is the highest, while the recall rate for other conditions are very low. The same can be found in *Figure 6.4* where the reference condition is "night-rain" and the highest recall rate happens at "night". The reason is the images



(a) accuracy (b) recall rate

Figure 6.1: Experiment 1 result

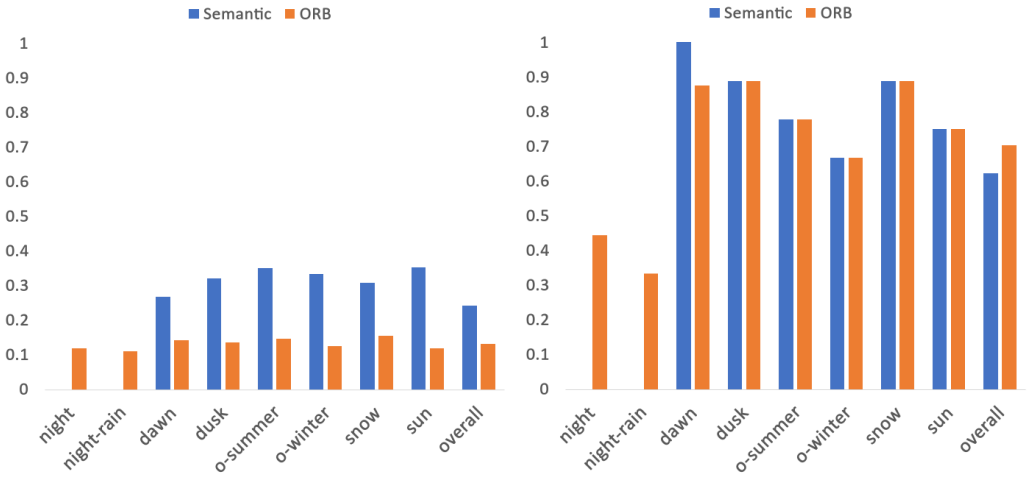


(a) accuracy (b) recall rate

Figure 6.2: Experiment 2 result: overall performance

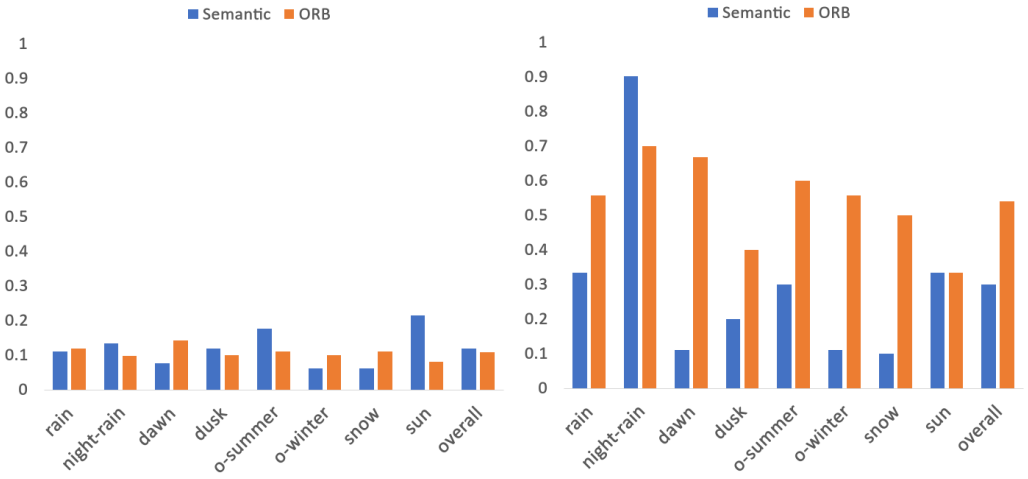
in both conditions are captured at night. Although the semantic images captured at "night" has very different appearance from semantic images captured in other lighting conditions, they have very similar appearance between each other. It doesn't mean they can be matched accurately since the accuracy is very poor. It just means they have similar feature distributions which are very different from feature distributions in other lighting conditions, although these features are not very identifiable.

From *Figure 6.5* it can also be observed that ORB features have 0 recall rate in many lighting conditions when the reference condition is "night-rain". This matches with *Experiment 1* in which the recall rate with ORB features is the lowest under "night-rain". It can also be found that overall recall rate with semantic features is also the lowest when reference condition is "night-rain". As explained above, the images in "night-rain" condition suffers from both poor lighting and glares from reflection of water.



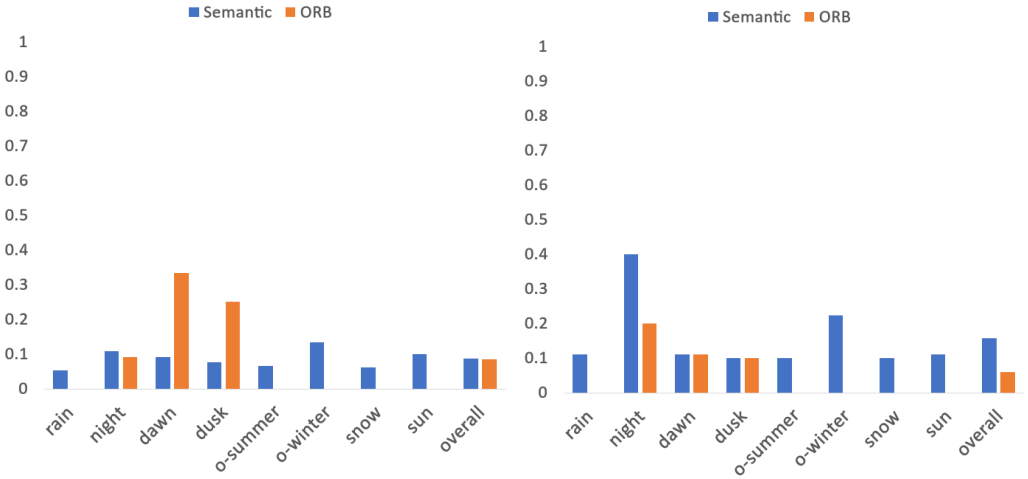
(a) accuracy (b) recall rate

Figure 6.3: Experiment 2 result: reference-rain



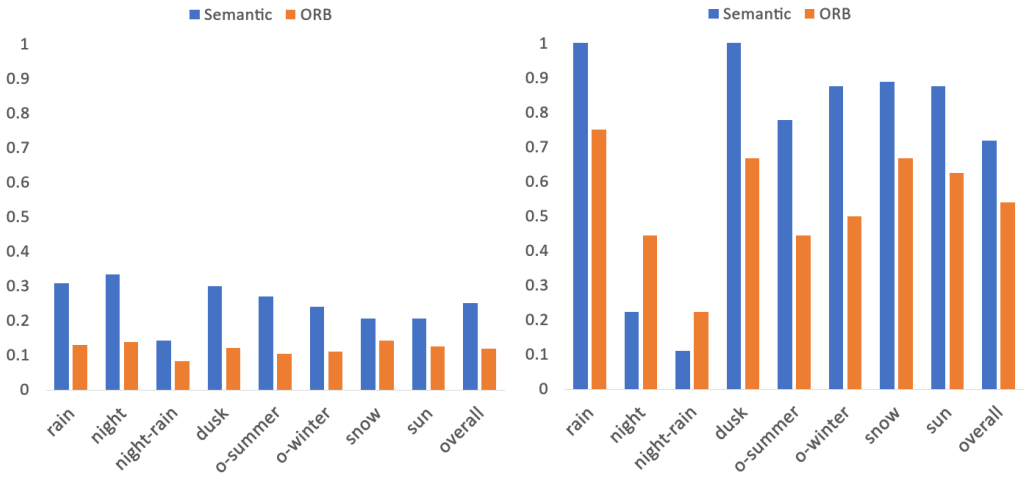
(a) accuracy (b) recall rate

Figure 6.4: Experiment 2 result: reference-night



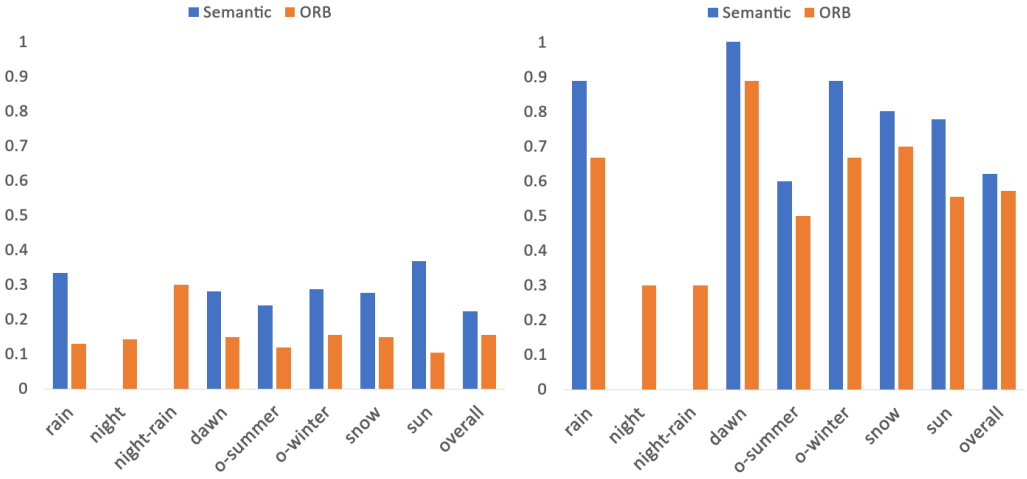
(a) accuracy (b) recall rate

Figure 6.5: Experiment 2 result: reference-night_rain



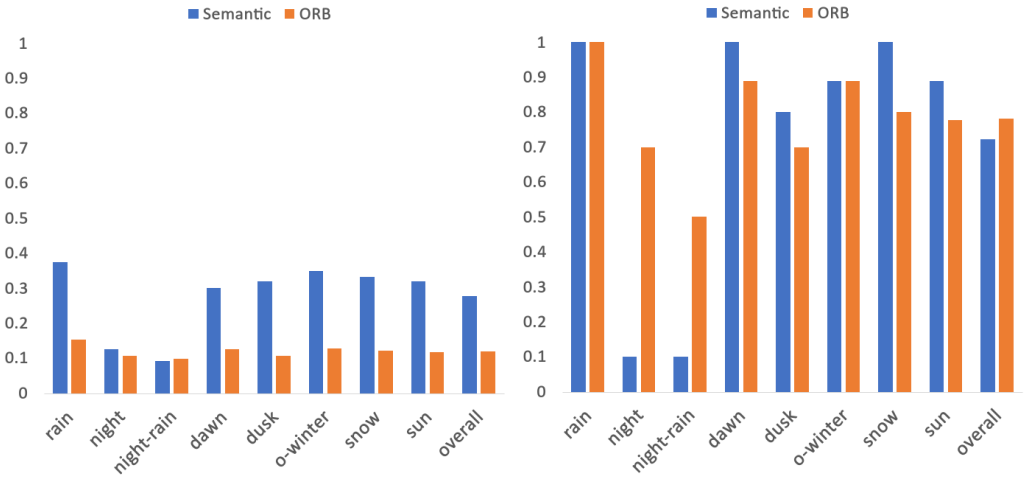
(a) accuracy (b) recall rate

Figure 6.6: Experiment 2 result: reference-dawn



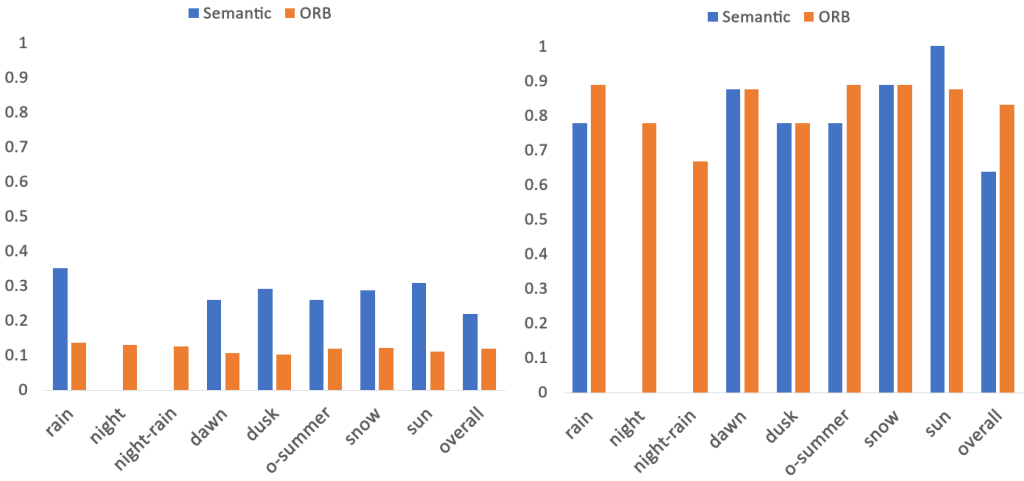
(a) accuracy (b) recall rate

Figure 6.7: Experiment 2 result: reference-dusk



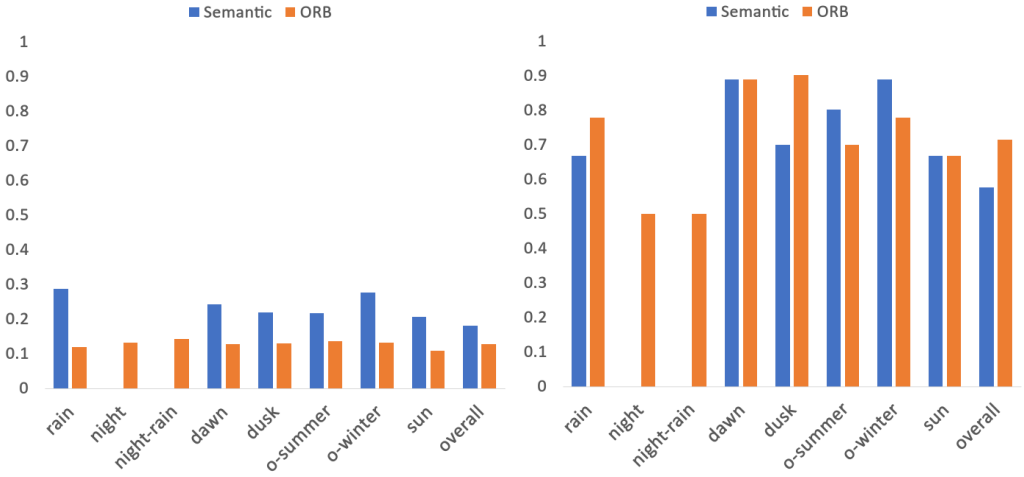
(a) accuracy (b) recall rate

Figure 6.8: Experiment 2 result: reference-overcast_summer



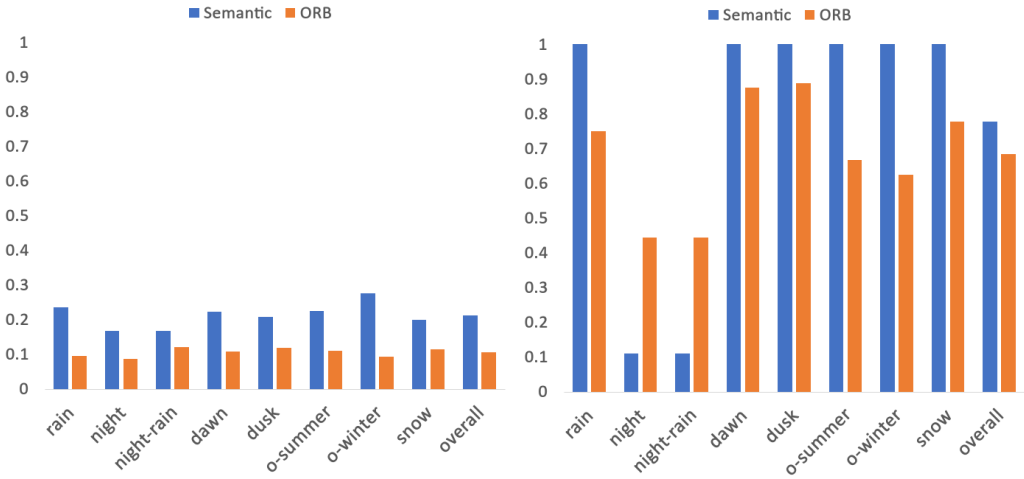
(a) accuracy (b) recall rate

Figure 6.9: Experiment 2 result: reference-overcast_winter



(a) accuracy (b) recall rate

Figure 6.10: Experiment 2 result: reference-snow



(a) accuracy (b) recall rate

Figure 6.11: Experiment 2 result: reference-sun

Chapter 7

Conclusion and Future Work

Based on the experiments carried out in this study, loop closure detection under different lighting conditions have higher accuracy and recall rate with semantic features than with ORB features. One exceptional case is when lighting condition change is too extreme that semantic segmentation becomes very unstable which makes semantic image quality too poor. On the other hand, the performance of loop closure detection under the same lighting condition is higher with ORB features than semantic features. A conclusion can be drawn that ORB features are more accurate under the same lighting condition, but semantic features are more stable under different lighting conditions.

For future study, the image similarity measurement algorithm developed in this paper could be implemented into ORB-SLAM2 system to test its performance in a complete SLAM system. Semantic images could be reproduced with a model trained with images at night such that semantic images captured at night would have better quality. Different template shapes and sizes could be tried to find out which template shape and size works better with the semantic template matching algorithm developed in this paper. The same experiments could be carried out with images in non-urban environments such as country-side where there might be less semantic features available.

Bibliography

[1] CityScape. Cityscape. `https://www.cityscapes-dataset.com/`. Accessed 15/09/2022.

[2] Dkaka. Deeplab_colab_demo. `https://github.com/Dkaka/Deeplab_colab_demo`. Accessed 15/09/2022.

[3] Rafiqul Islam and H. Habibullah. A semantically aware place recognition system for loop closure of a visual slam system. In *2021 4th International Conference on Mechatronics, Robotics and Automation (ICMRA)*, pages 117–121, 2021.

[4] kitti360. kitti360. `https://www.cvlibs.net/datasets/kitti-360/documentation.php`. Accessed 15/09/2022.

[5] Hongyang Li, Chengjun Tian, Lequan Wang, and Hongfu Lv. A loop closure detection method based on semantic segmentation and convolutional neural network. In *2021 International Conference on Artificial Intelligence and Electromechanical Automation (AIEA)*, pages 269–272, 2021.

[6] Will Maddern, Geoffrey Pascoe, Chris Linegar, and Paul Newman. 1 year, 1000 km: The oxford robotcar dataset. *The International Journal of Robotics Research*, 36(1):3–15, 2017.

[7] Yue Meng, Yongxi Lu, Aman Raj, Samuel Sunarjo, Rui Guo, Tara Javidi, Gaurav Bansal, and Dinesh Bharadia. Signet: Semantic instance aided unsupervised 3d geometry perception. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 9810–9820, 2019.

[8] Nathaniel Merrill and Guoquan Huang. Calc2.0: Combining appearance, semantic and geometric information for robust and efficient visual loop closure. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4554–4561, 2019.

[9] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE transactions on robotics*, 33(5):1255–1262, 2017.

[10] Torsten Sattler, Will Maddern, Carl Toft, Akihiko Torii, Lars Hammarstrand, Erik Stenborg, Daniel Safari, Masatoshi Okutomi, Marc Pollefeys, Josef Sivic, et al. Benchmarking 6dof outdoor visual localization in changing conditions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8601–8610, 2018.

[11] Yuxiang Sun, Weixun Zuo, and Ming Liu. Rtfnet: Rgb-thermal fusion network for semantic segmentation of urban scenes. *IEEE Robotics and Automation Letters*, 4(3):2576–2583, 2019.

[12] Wei Zhou, Julie Stephany Berrio, Stewart Worrall, and Eduardo Nebot. Automated evaluation of semantic segmentation robustness for autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 21(5):1951–1963, 2019.

[13] Maojing Zhu and Liang Huang. Fast and robust visual loop closure detection with convolutional neural network. In *2021 IEEE 3rd International Conference on Frontiers Technology of Information and Computer (ICFTIC)*, pages 595–598, 2021.