Deep Loop Closure SLAM

ROB 530 Team 2 Project Report
(Kun Sun, Ning Xu, Parth Chopra, Poorani Ravindhiran, Tribbi Kathuria)
{kirkun, nxu, parthc, poorani, tribhi}@umich.edu
October 1, 2019

Abstract—In this work, we evaluate a place recognition algorithm for loop closure detection, building upon the ORB-SLAM2 framework. We evaluate the performance of the proposed neural network generated HOG-like descriptors [termed Deep HOG] against ORB features with Bag-of-Words [BoW] descriptors and Histogram of Oriented Gradients [HOG] for performing place recognition and matching. The study seeks to improve on the loop closing architecture in ORB-SLAM2. Loop Closing is an essential component in SLAM that helps to create consistent environment maps and robot trajectories essential for long-term autonomy of mobile robots. Finally, we show that the global image DHoG descriptors perform better than ORB and traditional HOG descriptors in terms of accuracy and query time when evaluated over test datasets and show promise for being used for loop closure in a real SLAM system.

The Github repo for this work can be found at https://github.com/parthc-rob/deeploopclosure-slam. A video showing the results of our method can be found here, YouTube: Deep Loop Closure SLAM.

I. INTRODUCTION

Simultaneous Localization and Mapping [SLAM], as a problem domain, draws from diverse disciplines such as computer vision, machine learning, mathematical optimization, probability and statistics. SLAM applications include map building, object tracking, and egomotion tracking for mobile systems, each using a variety of sensors and motion models for specific problem domains[1].

The computer vision community has had significant contributions to the development of SLAM, warranting a novel subcategory called Visual SLAM [2]. Image processing and computer vision lends itself well to the SLAM problem as vision cameras provide a much richer source of information than sensors such as sonars, low-resolution LiDAR, Radar, IMUs, tactile sensors etc. As camera sensors as well as high-throughput compute has improved exponentially over the last few decades, both have improved to the point of being affordable and easily integrated with SLAM systems on mobile platforms such as ground / underwater / aerial vehicles, and various kinds of field-deployable robots.

An important module for a SLAM system is the Place Recognition module, which is responsible for recognizing if an area being observed has already been mapped in the trajectory of the robot. This module maintains a database of places in observed key frames to match a new entry against, which is then used for generating loop closure proposals. These loop closure hypotheses are used to globally optimize error over the trajectory and obtain accurate, consistent maps [3], [4], [5], [6].

A SLAM system works on a motion and an observation model of the system. However, due to uncertainties in environment and noise inherently present in the system, error gets accumulated over the course of time, which can specially become a problem when we want to map relatively larger areas. Such an error can be significantly reduced because with loop closure, the robot gathers more information about an already mapped environment, and uses that to optimize over the course of its trajectory, reducing the uncertainty in poses and the map [7]. A good place recognition module can prevent re-initialization of key frames in places that have already been mapped, which may otherwise be seen as new areas and be incorrectly mapped.

In a typical place recognition problem, an intuitive solution would be computing the similarity between two images using a distance metric [8]. Images obtained from camera sensors can be represented as a matrix with one or more spectral channels [R/G/B]. Upon experimentation, a metric purely based on image intensity values fails under
significant variations in viewpoint or illumination conditions.

Some methods focus on extracting features / key point descriptors from each image to build compact representations of the image, such as SIFT, SURF, BRIEF etc. These describe the local characteristics of individual patches, but different descriptors have performance trade-offs [9] between characteristics such as compute complexity, invariance to illumination / rotations, robustness to noise / occlusion / distortion etc.

Another aspect of prior computer vision research is to find a global representation of the image, termed a global descriptor. Bag-of-words [BoW] [10] approaches perform well in static scenes as a way to obtain global descriptor for each image. These depend on matching keypoint features in a query image to a reference vocabulary of feature words. There are also some other global descriptors which don’t rely on image features, and among them the Histogram of Oriented Gradients [HOG] descriptor gives a more promising solution for solving place recognition problem as it computes the image similarity score based on image pixel gradients and it is invariant to illumination condition changes, which is vital in outdoor scene recognition and closed loop detections.

Convolutional neural networks (CNNs) are a promising method of obtaining high-level image descriptors without hand-crafted features. They can be used as black-box image feature detectors for SLAM systems. However, these training based methods suffer from lacking of clean, labelled training data and limited computational power when required to perform inference on real-time deployments. They are also prone to overfitting on training data provided, in which case new test data produces unreliable outputs. Bulky CNNs with a large number of layers also make inference slow at the cost of learning more complex image representations and increased accuracy.

This work re-evaluates the efficacy of the place recognition module proposed in [11], implementing the deep neural network in a Keras framework. The proposed Deep HOG descriptor approach is then compared to the existing Oriented fast Rotated BRIEF [ORB] [12] + BoW approach in [13], the metrics being accuracy of place recognition and query compute time. Further, the results from the comparison justify integration with the existing ORB-SLAM 2 framework, to improve loop closing performance. The Deep HOG place recognition method is lightweight, fast and accurate, helping us simplify the loop detection module in the ORB-SLAM2, as illustrated in Fig. 1.

II. RELATED WORK

Visual Simultaneous Localization and Mapping (vSLAM) methods rely on camera for performing localization and mapping. Since camera is one of the cheapest sensor modalities and it gives rich sensor data of the environment, vSLAM methods have become popular over the years. Large-scale direct monocular SLAM (LSD-M) [14] and ORB SLAM [7] are two dominant vSLAM methods. Most of these are monocular SLAM methods, which perform SLAM on an image sequence obtained from just one camera, causing them to be scale independent. To overcome this major drawback, places visited already are detected using loop closure technique, the scale drift is estimated and then corrected. Monocular SLAM methods can either use the image intensities directly (like LSD-M SLAM) or key points of interest alone (like ORB SLAM) to estimate the location and surroundings. The first technique has less outliers as it retains a lot of positional valuable information, but has higher computational complexity and is invariant to changes in the scene. On the other hand, ORB SLAM methods overcome these challenges by using ORB features [15], which are rotation-invariant and are extracted from images quickly.

Other common feature descriptors used in the literature include Speeded Up Robust Features (SURF) [16] and BRIEF. Even though SURF descriptors are robust to scale variations, they require more computation. SURF descriptors are memory-heavy, slowing down query and making them unsuitable for fast place recognition. But, all the dimensions of SURF descriptors may not be needed for matching. Hence, several methods like Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA) and Locality Sensitive Hashing (LSH) have been used to convert these high-dimensional descriptors to binary strings, as shown in [17].

These descriptors still need to be computed, and are memory-intensive. BRIEF descriptors solved this problem by computing the binary strings directly without finding the descriptors, making the process of matching much faster. It also provided high recognition rate except when there was large in-plane rotation.

One of the earliest place recognition methods used in vSLAM was Fast Appearance-Based Mapping (FAB-MAP) [18], which uses a probabilistic approach to do place recognition based on the appearance of places. A generative model for the bag-of-words (BoW) data, which is capable of identifying common objects without supervision and rejecting false matches due to perceptual aliasing, is learnt. This method is computationally efficient. BoW model refers to a way of extracting features from image data, which basically is another way of representing the same data by describing the occurrence of ‘feature words’ in the data. The BoW model is created in two steps: a vocabulary of known words is created and a score is assigned to each of the words in the vocabulary, which could be a simple binary indicator of its presence or the frequency of it.

Following this, a novel method using bag of words obtained from FAST + BRIEF features [19] was used for place recognition. This was the first time a vocabulary tree
that discretizes a binary descriptor space was built and used to speed up the process of finding correspondences. But, since BRIEF descriptors are not rotation and scale invariant, this method was limited to scenes taken from the same viewpoint in planar paths. This drawback was overcome by building a vocabulary tree based on ORB features, which is used in the place recognition module of the traditional ORB-SLAM framework against which we compare our results.

Recently, ConvNet based supervised approaches have been used for place recognition. In [20], the Overfeat network is trained on the ImageNet dataset, which consists of 1.2 million images and 1000 classes, to give a deep representation of the input image. Place recognition is then performed by comparing these feature vector representations of different images. Even though these large representations achieved a 75 percent increase in recall at 100 percent precision, significantly outperforming all previous techniques, these hindered their usage for real-time operation. Similarly, Sunderhauf et al. [21] used ConvNet features to match sub-regions corresponding to landmarks with an aim to get better descriptors, yet again they could not be deployed for real-time operation. In [22], to do cross-season visual place recognition, a method is proposed to extract the full output of an intermediate layer of the network and build a lower-dimensional descriptor by omitting the activation of filters corresponding to season changes. These methods relied on either large labelled relevant training data or generic pre-trained neural networks that were not specifically trained for loop closure.

Unsupervised deep learning architectures are being widely deployed nowadays. In [23], a stacked de-noising autoencoder (SDA), which autonomously learns a low-dimensional representation from input data in an unsupervised way, is discussed. One major drawback of this technique was that the network was trained to reconstruct images whose pixel intensities were randomly altered, instead of focusing to make the method invariant to variations that could arise because of viewpoint changes as all random variations are not natural. This issue is addressed in [11] by introducing randomized projective transformations to emulate natural viewpoint changes instead of random variations in pixel intensities. In addition, HOG descriptors are used to encode the images to achieve illumination invariance along with viewpoint invariance.

III. METHODOLOGY

A. Unsupervised Convolutional Neural Network

In our project, we apply a lightweight Convolutional Neural Network (CNN / ConvNet) for loop detection [11]. This approach takes advantage of the ConvNet architecture to map high-dimensional input images into a low-dimensional descriptor space, and then matches the descriptors according to their similarities. Though the most popular Bag of Visual Words [19] approach also has a similar two-step structure of extracting descriptors and matching, ConvNet based methods extract deeper level of descriptors for more accurate matching results. The disadvantages usually faced while using such deep networks include high computational complexity and large memory usage, causing challenges for real-time implementation. On the other hand, a lightweight autoencoder [24] structure, which is much more computationally efficient than many state-of-the-art very deep neural networks such as ResNet [25] and InceptionV3 [26], is used in our model. It also provides reliable accuracy if it is specifically tailored for the place recognition task. Therefore, the following approaches are taken to train the network to encode robust descriptors under different appearance conditions:

- As mentioned in II, the HoG descriptor has the property of illumination invariance as it takes image gradients into account. To make use of this advantage, the loss function of the autoencoder is defined as the euclidean distance (L2 norm) between the HOG descriptor computed using OpenCV directly and the reconstructed Deep HOG descriptor.
- One vital drawback of HOG descriptor is that it is not invariant to viewpoint changes. To overcome this disadvantage, random projective transformations are applied on the training images to make DHoG invariant
Due to the above two improvisations, the computed descriptor will be both illumination invariant and viewpoint invariant. These properties are essential for using in a real SLAM system, especially in outdoor scenes.

Data flow in the training pipeline is illustrated in Fig. 2.

- Input RGB image is first converted to gray scale and re-sized to 120x160 size.
- Random projective transforms are applied on this image, by performing the \textit{add-noise} step in the autoencoder [24].
- One amongst the transformed and original images is randomly picked and used as the training image.
- The conventional HOG descriptor is computed on the other image (that was not randomly picked in the previous step) to use as ground truth.
- The training image (rescaled by 1/255) is fed into the ConvNet to attempt to reconstruct a HOG-like descriptor in forward propagation.
- The ConvNet is trained by minimizing reconstruction loss between descriptors obtained using these two different methods.

### B. Network architecture

The input images are padded using valid padding technique, which drops the right-most columns (or bottom-most rows alone. Then, these padded images are fed to a 2-dimensional convolutional layer with 64 filters, each having a kernel size of 5x5. The filter stride dimension is set to 2 and 2-D max pooling of kernel size 3x3 is applied to reduce the dimensionality of the input images. This is followed by a normalization layer, where local response normalization (LRN) is performed. This causes a kind of lateral inhibition by normalizing over local input regions. Another padding layer and a 2-dimensional convolutional layer follow this. This convolutional layer has 128 filters of kernel size 4x4 and is followed by max pooling of kernel size 3x3. The output of this is fed into a third convolutional layer which has 4 filters of kernel size 3x3. All the convolutional layers have Rectified Linear Unit (ReLU) activations. The output of the last convolutional layer is fed into two fully connected layers with sigmoid activation to give the final feature descriptor.

### C. Network training

We build our model in Keras [27] and use 6 million training images from the Places2 dataset [28]. The learning rate is 0.0009, and the momentum is 0.9 with a decay of 0.0005. We set the batch size as 256 and feed 3906 batches of images in each epoch. We trained nearly 20 epochs for our final model.

### D. Loop Closing

The proposed method allows simplification of the existing loop closing module in the ORB-SLAM2 framework, wherein instead of using the traditional BoW representation, we use the DHoG descriptor obtained from the unsupervised convolutional network discussed in the previous section. The main steps involved in the algorithm can be summarized as follows:

- Query existing database to compute similarity score and the corresponding keyframe id
- Reject queries where the similarity score is less than the preset threshold.
- Return the loop candidates to be checked for consistency over 3 queries
- Compute the transformation between the loop candidates and the current keyframe
- Apply the same transformation to the next frame and the compute the optimization again
- If we get enough inliers the loop is accepted

Rest of the processing lies in fusing the loop candidates, the computed transformation (that reduces the re-projection error) is propagated to all previously observed keyframes and if the transformation found enough inliers the points were fused, giving us more accurate maps [7].

### IV. Evaluation and Results

#### A. Accuracy Evaluation

To analyze the performance of our place recognition module, we use standard evaluation tools including confusion matrix and precision-recall (PR) curve. The performance of a place recognition system is determined by how well the estimation of similarity between the compared images coincides with the ground truth about similarity between these images. To evaluate this, we use the Campus Loop Dataset.

![Fig. 3](image-url) This figure shows two images from the Campus Loop Dataset, showing the same place under different lighting, seasons, and camera pose. The HSV heatmap shown compares the reference model obtained from [11], with the custom-trained model in our Keras implementation. We see that our trained model empirically performs well compared to our reference.
Fig. 4. Based on Deep HOG descriptor similarity, a score between a query keyframe and prior keyframes in a trajectory are calculated. We see that the highest score appears at image 042, which is then assumed to be the matched image. This is compliant with our ground truth.

dataset[11], which consists of two sequences of 100 images each, taken with extreme weather and viewpoint variations, as well as the well-known KITTI sequences.

1) Confusion Matrix: The Deep HOG [DHOG] descriptor \(d(i)\) for each image \((i)\) is, then, computed by performing forward propagation using our trained model. The similarity score for each image pair in the database is computed as the dot product shown below.

\[
S_{ij} = d_i \cdot d_j
\]  

This product is, then, normalized to a range between 0 and 1. A higher similarity score means this image pair is more likely to be considered as a matched pair and hence, it forms a loop closure candidate. We define a confusion matrix, whose element \(i, j\) is defined as the similarity score between the images \(i\) and \(j\).

We compute the confusion matrices using the score from ground truth(1 for a match and 0 for others), our DHOG results and the DBoW2 representation with ORB features. Figure 5 displays the plots for these confusion matrices. A darker color refers to a higher similarity score. We can observe that the confusion matrix obtained using our model coincides with that of the ground truth. In addition, there are additional false positive detection in the confusion matrix computed using the traditional method with DBoW2 library or DHOG descriptor dot product. Lowering the threshold makes the system more liberal in reporting loop closures, leading to higher recall but correspondingly low precision (loads of false positives).

Figure 6 shows the PR curves for our method and the traditional method which uses BoW representation and ORB features. The AUC is higher for our method, thus showing its effectiveness over the other method compared.

\[
P = \frac{\text{Number of correctly detected loops}}{\text{Number of reported loops}}
\]  

\[
R = \frac{\text{Number of correctly detected loops}}{\text{Number of loops in the ground truth}}
\]

The higher the values of precision and recall, the higher is the accuracy of the method. The area under curve (AUC) is an important way of evaluation. Higher AUC is desired. In our case, we explored the PR trade-off by changing the threshold value applied to the score computed by DBoW2 library or DHOG descriptor dot product. Lowering the threshold makes the system more liberal in reporting loop closures, leading to higher recall but correspondingly low precision (loads of false positives).

2) Precision-Recall Curve: We also utilize the precision-recall curve, a standard method to evaluate prediction accuracy, to quantify the effectiveness of our model. By definition, recall(\(R\)) expresses the ability to find all relevant instances in a data set and precision(\(P\)) expresses the proportion of the data instances which are actually relevant to data instances which the model calls to be relevant. These are represented in the following equations:

\[
P = \frac{\text{Number of correctly detected loops}}{\text{Number of reported loops}}
\]  

\[
R = \frac{\text{Number of correctly detected loops}}{\text{Number of loops in the ground truth}}
\]

Fig. 6. Precision-Recall (PR) curve- Our method (upper curve) outperforms the traditional BoW method (lower curve)

B. Efficiency Evaluation

As we proposed before, we want to integrate this DHOG descriptor into state-of-the-art SLAM system like ORB-SLAM2. To achieve this goal, it is essential for us to evaluate the efficiency of this descriptor based detection method and make sure it is fast enough for a online SLAM system. We perform run time evaluations of both average descriptor computing time and total database querying time over different image database size for a single-nearest neighbor search.
In this experiment we use HOG descriptor and bag of words of ORB features as two benchmarks since HOG descriptor represents the traditional computer vision method based on OpenCV library and DBoW2 with ORB features is one of the fastest place recognition libraries used in many SLAM systems, and compare them with our results. It is worth noting that though the run time may vary with different hardwares but the relative relationships between different descriptors remain the same. We test all the run time results using the same device without GPU acceleration and all the descriptors are selected to have the same size.

Figure 7 shows the results of the descriptor extraction run time comparison, obviously our DHOG descriptor is faster than the DBoW2 with ORB features one. For each DHOG descriptor, we performed one forward propagation using our well-trained model on the Keras. In terms of descriptor extraction, HOG descriptor is even faster, though it is not able to maintain a good matching accuracy.

Figure 8 shows that our method achieves a better efficiency when querying a large database compared with DBoW2. It was observed that there are not enough images in Kitti sequences so we find the other irrelevant image datasets and build the database over them. DHOG and HOG descriptor are supposed to have the same query time as they are using the same distance metric to find a match.

C. ROS 2D Simulation

Fig. 9. 2D Simulation of DHoG detection method over KITTI Sequence 00. Black line shows the robot ground truth trajectory and the teal point indicates there is a pair of matched keyframes.
In addition, we perform a 2D simulation using the RViz visualization tool in ROS. In this simulation, we integrate our Keras model to compute DHOG descriptor and make the assumption that we will take one keyframe after every seven frames. The robot pose is obtained from the KITTI dataset ground truth and a linear search over the whole database has been implemented to find the match or call it the nearest neighbor. From Figure 9, we can see most of the loops are detected correctly, which are marked in teal and there isn’t even one false positive loop in our result, which indicates the potential of implementing this loop closure detection method with a real SLAM system.

V. CONCLUSION

In this project, we proposed an effective visual Loop Closure Detection method based on an unsupervised convolutional neural network, which was proven to be effective on a wide range of benchmark data sets without any re-training based on the network proposed in [11]. The novelty of this study lies in recognizing the applicability of this recent technique in computer vision and deep learning to improve loop closing in the state-of-the-art SLAM implementation, which is ORB-SLAM2.

The suggested neural network model is lightweight and allows fast real-time operation that is essential to the working of a SLAM system. Moreover, the proposed loop closure detection method is more accurate in the sense that it can detect similar images despite variations in viewpoints, illumination conditions or occlusions. The accuracy and efficiency of the place recognition system motivates its use in the ORB-SLAM2 framework, that handles frame tracking, local mapping, loop closing and re-localization in parallel.

Future work in this direction would involve fine tuning of the parameters, that describe the threshold of the similarity score of two key frames, used in reliably finding the loop closure candidate frames, in order to further improve the accuracy. Giving the threshold a low value means the method will optimize and check for outliers in more adjacent frames, thus increasing the accuracy but negatively affecting the computation time, questioning its application in real time. On the other hand, a higher threshold means the method might end up rejecting even the good loop closure candidates. As a result, the computation time required will decrease but at the cost of reduced accuracy. Hence, it is important to maintain this trade-off carefully and tune the threshold parameter by a simple linear search or by a learning method. Other potential directions of work include the evaluation of scenarios inducing high tracking loss and performing global bundle adjustment on the detected loop candidates.

REFERENCES


[27] François Chollet et al. Keras. https://keras.io, 2015.