CS393R - Autonomous Robots **Project Report, Fall 2021**

Correlative Scan Matching using Convolutional Neural Networks

January 16, 2023 RJ Antonello, Sai Shankar N, Sharachchandra Bhat

1 Introduction and Literature Survey

Simultaneous Localization and Mapping (SLAM) plays an important role in various fields, such as autonomous driving, indoor navigation and virtual/augmented reality. Because of its elegant rasterized look-up feature, Correlative Scan Matching (CSM) has gained a lot of traction as a SLAM front-end alternative for indoor navigation. But, CSM does an exhaustive search over a 3-D grid of poses (x, y, θ).

In this project, we propose a solution that is based on regression, rather than an exhaustive search. We propose a Deep Neural Network (DNN) based solution, that employs convolutional layers, to regress the relative pose transformation between two laser-scans.

Our idea is based on the premise that given two local occupancy maps belonging to two different robot poses, the convolutional layers in the DNN can capture the correlation between the occupancy maps. This approach is similar to Deep Visual Odometry papers that have been published recently. In general, all these works exploit the global feature correspondences through multiple convolutional layers to regress the relative pose transformation.

Additionally, there have been similar works on finding correspondences between two LiDAR scans. The most similar approach to ours that we could find is [1]. [1] uses convolutional layers to find correspondences between multiple range values of two laser scans. But, this is an inefficient way to use convolutional layers. Hence, we suggest the use of occupancy maps, similar to the rasterized cost maps used in CSM.

A key advantage of such a setup is that, there is no necessity to tune motion model, observation likelihood model, etc. By just providing the occupancy maps, blown-up with the robot safety radius, we hope the DNN parameters can learn the implicit error in motion model and observation likelihood model for various scenarios.

2 Problem Formulation

In this section, the proposed DNN model for SLAM front-end is explained in detail. It mainly consists of repeated convolutional layers to capture feature correspondences in a global manner. The proposed DNN model can be considered to compute the conditional probability of relative pose transformation between two robot poses, given the two corresponding occupancy maps built using laser scan.

$$p(T_{t,t+1}|X_t, X_{t+1}; \theta) \tag{1}$$

where $T_{t,t+1}$ represents the relative pose transformation between the two robot poses, X_t and X_{t+1} are the occupancy maps at time-steps t and $t+1$, and θ represents the DNN parameters.

The modelling and probabilistic inference are performed in the DNN. To find the optimal parameters θ^* for the system, the DNN maximises

$$\theta \approx \mathbf{argmax}_\theta p(T_{t,t+1}|X_t, X_{t+1}; \theta)$$

To learn the parameters θ of the DNN, the Euclidean distance between the ground truth pose $T_{t,t+1}$ and its estimated one $\bar{T}_{t,t+1}$ is minimised.

3 Data Collection

In this section, we explain our data collection and pre-processing pipeline.

- Traverse a pre-defined path in UTAutomata simulator or using the F1/10th.
- Store the laser scan and the corresponding robot location at every time-step.
- Create a local occupancy map for each laser scan assuming the robot to be at the center of 224 x 224 grid which spans the area of 15m x 15m
- Generate possible combinations of pairs of local occupancy maps and the corresponding relative pose transforms, such that there is considerable overlap between the observations of the two time-steps.

4 Network architecture and training

We explain our training and inference pipelines, and the network architecture in this section. A training set and test set is built from separate robot trajectories to ensure proper generalization of the network. A convolutional neural network model closely related to the ResNet architecture was used. This is a five-layer CNN which has BatchNorm and ReLU layer in-between each convlutional layer. These five convolutional layers are followed by four additional linear layers that are also interleaved with ReLU layers. A full description of this network architecture is given in Table 1.

Table 1: Network Architecture

Layer	**Receptive Field Size**	**Padding**	**Stride**	**Number of Channels**
1	5	0	2	16
2	5	0	2	32
3	5	0	2	64
4	5	0	2	128
5	5	0	2	256
6	FC	FC	FC	4096
7	FC	FC	FC	256
8	FC	FC	FC	16
9	FC	FC	FC	1

Training proceeded by feeding individual backpropogating over pairs of images generated as described in Section 3 until convergence. Convergence was defined by reaching the maximum

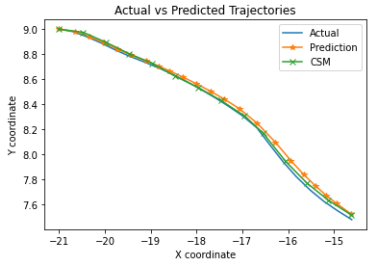
performance as determined via measuring mean-squared error loss against the objective on our held-out test set. Only models that improved performance on this test set were considered to have improved.

5 Experiments

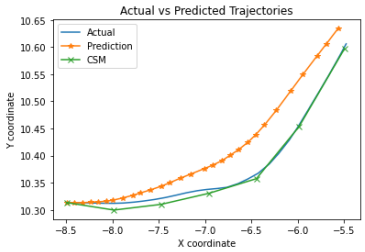
We have collected upto 10 trajectories using the UTAutomata simulator, and generated around 100,000 pairs of local occupancy maps for training the DNN. We compare our proposed method against the ground-truth trajectory and report two metrics, relative drift in the translation distance ($\sqrt{\Delta x^2 + \Delta y^2}$) and angular distance ($\Delta\theta$) throughout the test trajectories, and at the end of the test trajectories. Additionally, we show a comparison against CSM. We aim to get results as close as possible to CSM, as CSM does an exhaustive search on the possible set of relative pose transformations.

Table 2: Quantitative Results

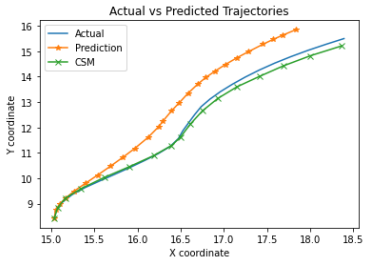
Test Trajectory	**RMS Error** (m, deg)		**End of Trajectory Error** (m, deg)	
	DNN	*CSM*	*DNN*	*CSM*
1	$(0.09, 1.2)$	$(0.06, 0.6)$	$(0.04, 1.3)$	$(0.08, 0.6)$
2	$(0.16, 2.1)$	$(0.03, 0.6)$	$(0.09, 4.6)$	$(0.04, 0.9)$
3	$(0.45, 6.6)$	$(0.25, 3.2)$	$(0.66, 8.1)$	$(0.24, 0.1)$



(a) Trajectory-1



(b) Trajectory-2



(c) Trajectory-3

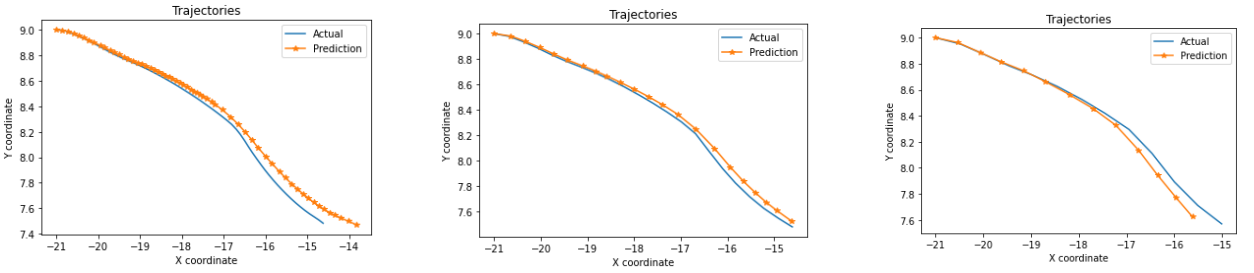
Figure 1: Qualitative Results

6 Conclusion

1. We can see that even when the predicted trajectory diverges from the ground truth, it is similar. The divergence is due to the build up in the prediction errors of relative poses and the shape of the trajectory is maintained because the prediction errors are not too large.

2. An important point to note is that our input space is extremely sparse. There are not too many discriminative features that provide a cue to regress the relative pose transformation. Hence, generalization is a major issue. This can be observed in the divergence in test trajectory predictions.

3. It was observed that when the distance between two successive poses, for which the network made an inference, varied so did the predicted trajectory (2). This is an interesting observation that merits further investigation.



(a) Successive pose distance of 0.1m

(b) Successive pose distance of 0.3m

(c) Successive pose distance of 0.5m

Figure 2: Dependence of trajectory on distance between successive poses

7 Future directions

There is a lot of flexibility with regards to how the trajectory data is provided to the network during training. Finding a method that would work well with fewer trajectories can be fruitful.
A direction for immediate future work would be to train and evaluate on many more varied trajectories and environments. The generalizability should naturally follow with a diverse and rich dataset.
Continuing along the same lines, this method can be extended to real world scenarios. The two major tasks would be to collect data of the environment with a 2D Lidar Scanner for training and write the algorithm to run in real time. Incorporating this with the SLAM optimizer would provide an end-to-end solution to the SLAM problem.
Another track of work would be to train the network in an unsupervised way. This would mitigate the need to label real world data.

8 Links

`https://github.com/sharachchandra/scan-matching-cnn.git`
`Project Data`

References

[1] J. Li, H. Zhan, B. M. Chen, I. Reid, and G. H. Lee. Deep learning for 2d scan matching and loop closure. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 763–768, 2017.