A Transformer Approach for Camera-to-LIDAR Data Registration

Ju Wang*, Yong Tang, Venkat R. Dasari‡, Billy Geerhart ‡, Brian Rapp‡, Peng Wang‡, Wei-Bang Chen*, Isaac Watts*
* Virginia State University, Petersburg, VA, USA
‡DEVCOM Army Research Laboratory, Aberdeen Proving Ground, MD, USA

***Abstract*—We proposes a novel method for camera-to-LIDAR calibration, which estimates the relative transform from LIDAR to camera sensors during on-field run-time without specialized equipment. The proposed method trains a transformer model to directly predict the 6D pose between the camera and LIDAR. Our method uses vision transformers to extract features from the LIDAR and camera image and calculate the similarity for candidate pose hypothesis. The results demonstrate the superiority of the proposed method over baseline approaches, showcasing its potential for practical applications.**

***Index Terms*—Robotics, LIDAR, calibration, edge computing, 3D object detection**

I. Introduction

There is a growing need to register and render 2D imagery data and 3D point cloud data from different sources in an accurate manner. At the core of the problem is to establish 2D pixels and 3D points mapping under unknown spatial relationship between two sensors. This capability is important for tasks such as robot team exploration, mapping, and data visualization. As such, an efficient and robust algorithm to calibrate external parameters is required to diffuse data from multiple sensor sources. Such algorithms, often implemented via a parametric estimation process, are critical to accurately "superimpose" synchronized data streams such as the camera data and the LIDAR point cloud.

An example work is the fusing of the LIDAR data and camera data by Wang et.al., [1], which uses local connectivities in the image domain to fine-tune the associations between the LIDAR points and image pixels. For these systems to be successful, one often take-for-granted an unspoken assumption: the spatial relation between the LIDAR sensor and the camera sensor is known. However slight deviation in the transformation parameters can result in significant mislabeling of the sensor data, which make it necessary to re-calibrate the inter-sensor transformation parameters.

In this work, our focus is to use a deep learning technique to estimate the camera-to-LIDAR displacement. The general problem is: given two modalities of data source, how do we train a deep neural network to estimate the relative pose? An intuitive idea is to utilize a supervised learning scheme to feed the network with image and LIDAR data to generate a target 6D pose. This would require the network to directly learn the features of the 3D point cloud and 2D image, which is a difficult task to train.

Nevertheless, many have observed that the two data modalities are strongly correlated since they are observed by two sensors in close vicinity. In particular, 3D LIDAR data projected to an image plane bear many similarities to the camera-observed RGB data, and hence allow some features to be comparable to the image features. Features observed in one can be easily found in another, albeit in different representations. Our method is inspired by recent success in domain adoption such as Domain Adversary Neural Network (DANN), where a domain "invariant" image encoder is trained to learn image features persist across different domains. Recent success in Contrastive Language-Image Pre-training (CLIP) [2], [3] even shows that two vastly different domains, natural language description, and image domain, can be trained to produce similar embedding. We hypothesize that domain adoption can be extended across different sensing modalities, such that a similar feature embedding could be learned for a point cloud encoder and an image encoder. Hence our approach is to train a network to learn cross-domain embedding/features from the two relevant sensor domains: the projected LIDAR point domain and the 'normal' image domain. Following these observations, our overall method is described below:

1) Compute the LIDAR points projection based on a set of rotation angles.
2) Computer the embedding for both LIDAR projection and camera image.
3) Compute the similarity of the projected depth images and the camera image.
4) Train the LIDAR-Image-similarity network by back-propagating the similarity score
5) At inference time, the 6D pose that produces the highest similarity score is selected.

We evaluated the proposed method using a combination of simulation data and field data collected by a fully loaded Clearpath Jackal robot. Our implementation and computing pipeline is optimized to achieve real-time performance on resource-constrained edge computing autonomous platforms such as unmanned ground vehicles (UGVs).

II. Related Work

The camera-to-LIDAR calibration is closely related to the Perspective-n-Point problems [4], [5], [6]. The problem is also studied in camera pose estimation [4], also referred to as viewpoint recover problem [5]. Conventional solutions

2835-5776/24/$31.00 ©2024 IEEE
DOI 10.1109/IRI62200.2024.00072

for these problems solve 2D-3D correspondence through an iterative optimization method. In the case of P3P where exactly three point pairs are given, a close-form solution can be obtained [4].

A closely related problem is to estimate rigid motion using two sets of 3D point clouds. Since Besl's seminal ICP paper [7], there is a rich collection of improvements in recent decades [8], especially with the availability of high-resolution 360-degree LIDAR sensors. The main idea in ICP and its variant is to gradually recompute the point correspondence in such a way that a better transformation matrix can be computed. Our method borrows a similar strategy to assure the convergence of the solution.

In our previous work[9], an Iterative Closest PnP (ICPnP) was proposed to solve the calibration problem using a common foreground object. The problem was formulated as a bundle adjustment problem between the discrete-time data and LIDAR data from the same robot. The disadvantage of this method is that it needs a separate process to extract 3D points corresponding to the front object, which impose an additional computation cost.

On the other hand, a growing trend to handle multi-modality data scenarios is to utilize deep learning models for sensing and scene understanding[10].

Cross-domain training has gained a lot of tracking recently to bridge the gap between different domains. In [3] OpenAI used contrastive pre-training to learning multi-modal embedding space between text and image. Within the image domain, Domain adversary training [11] shows the feasibility of learning invariant features in general. Recent variations of DANN have tried to extend the method to multiple domains and applications in generative networks.

III. Our approach

A. TR Parameter Estimation

Without loss of generality, we assume the LIDAR as the 'parent' frame and the camera frame is the child. The goal is to estimate a transformation matrix TR^c such that for any point $p_l = (x, y, z)$ in the LIDAR frame, its projection in the camera plane can be estimated as: in equations (1-2).

$$I_c = M \times TR_c \times p_l / z_c \tag{1}$$

$$p_l^* = TR_c^{-1} \times M^{-1} \times (z_c . I_c) \tag{2}$$

Here p_l is the 3D point cloud in the LIDAR frame, p_c is the poiint in the camera frame, and I_c is the projected pixel set in the image plane. We assume that the camera projection matrix M is known. The 3D transformation matrix $TR_c = [R\ |t]$ has two parts: R is a 3x3 rotation matrix and t is a 3x1 vector for translational offsets. The translation vector t coincides with the origin of the LIDAR frame in the camera frame.

If a set of 3D point sets P and the corresponding image point set I is given, the Tr solution can be estimated by a mean square error (MSE) estimator. In our problem, we don't assume any prior knowledge of some 2D-3D point correspondence, hence the conventional method would not be applicable. Furthermore, an immediate challenge is to define the measurement to evaluate the quality of an Tr estimation, which is necessary for any optimization problem. Two possible metrics are possible : (1) a function to measure the closeness of the projected LIDAR image and the camera RGB image, or (2) a function to measure the fitness of the LIDAR points inversely projected from the camera image. Here we choose the second metric, details will be discussed later. With such, essentially we try to optimize

$$argmin_{T,R} \sum_{i \in I} ||d(M * TR)^{-}1(i), P)||$$

Here we use $(M * TR)^{-}1(i)$ to represent the 'inverse' transformation of an image pixel back to $p \in P_l$ under given calibration parameters TR. Notice that the matrix notion here is not to be taken literally. $d(p, A)$ is the distance of p to a 3D point set.

TABLE I
SYMBOLS DEFINITION

$P_l(t)$	True PC at t in robot lidar frame
(x_c, y_c, z_c)	3D point in camera frame
$\bar{s} = (x, y, z, \bar{q})$	object position and quaternion in lidar frame
$I_o(t)$	Object 2D image at t
$P_o^*(t)$	predicted Object PC at t
TR_c	TF matrix to cam frame
M	2D cam proj matrix

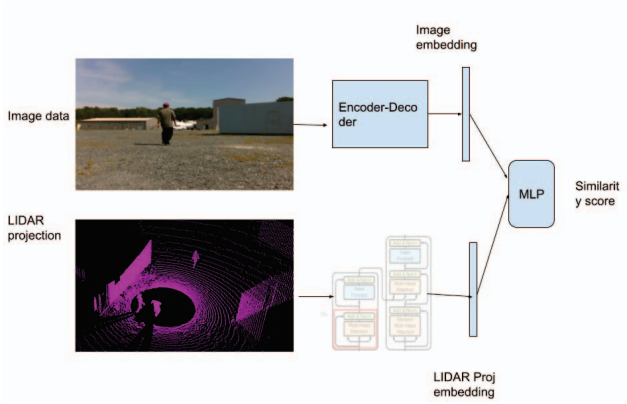

Fig. 1. Computing flowchart of the Lidar-Image Similarity Network. Two transformers are used to extract features for LIDAR projection and camera image. The network outputs a similarity score between the two input images. Transformer diagram from [12]

B. TR estimation algorithm description

Instead of directly estimating the transformation parameters using a deep neural network, our strategy is to use a deep neural network as a discriminator to 'compute' the similarity of the LIDAR projection and the camera image. The structure of the proposed network is shown in Figure 1. The expected behavior of the network after training is:

- the network output a single scalar value (0,1) to indicate the probability of the LIDAR projection being the 'same' as the camera image

- The similarity score should be proportional to the distance between the LIDAR projection and the camera image.

We use a similar strategy as the one used in LCDNET [13] to generate 6D pose proposals: The candidate pose (TR) is first generated by a rotation matrix uniformly sampled in the RPY axis. To accelerate the network training, we cap the maximum rotational offset to 40 degrees. A linear translation offset is then applied.

Input:
PC: point clouds at two time inst;
I_0, M_0: Image and 2D object mask segmentation ;
TR_0: initial calib matrix;
Output:
TR_n: final calib matrix;
Function `CaliberationOptimizer`
`/* Extract object points from PC0 */`
$(Tr_1, ..., Tr_N) \leftarrow$ (N) */
`/* /* Compute Lidar Projection */`
$(\hat{L_1}, L_N) \leftarrow$ `Projection` (TR_i, P_0) */
`/* /* Forward: Lidar Projects,` I_0 `*/`
for $i \in (0, 1, 2, \ldots)$ **do**
$I_k \leftarrow (\hat{I_0}, L_i)$ */
`/* Apply` (TR_k, P_0) */
`/*` $Score(I) \leftarrow TRdist(\text{Tr}_i, M_0, PC)$
end
$Tr^* \leftarrow$argmax$_T r$ {Score}
end
Algorithm 1: Lidar-Camera TR Parameter Search

C. Feature Extraction and Loss function

The loss function is calculated in a similar way as [3]. The embedding computed by the LIDAR projection head and the image head are directly input to calculate cosine similarity, as shown in Figure 2. For LIDAR projection generated with ground truth Tr, the maximum cosine similarity is to be expected (in the matrix diagonal) and used as a positive training dataset. None-diagonal data pairs are treated as negative data points.

Let L and I represent the embedding vectors computed by the two encoder heads. The image encoder used is a vision transformer as [14], [3]. The initial parameter of the image head is pretrained with ImageNet. The LIDAR projection head uses an identical network structure but is initialized with random weights. The loss function is defined as

$$Loss = 1 - \frac{L \cdot I}{||L|| ||I||}$$

D. TR evaluation metric

Given an estimate Tr, an image pixel $(u_i, v_i) \in I$, we can find its back projection in the 3D space by drawing a ray from the (estimated) camera focal point through (u_i, v_i) (see eq 2). The closest 3D Lidar point to the imaginary ray is the 3D point $p_l(I)$. Thus a simple way to evaluate the Tr matrix is to simply

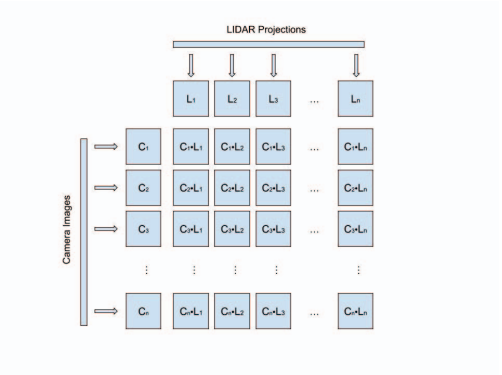

Fig. 2. Loss function: the loss matrix between the LIDAR projection embedding and camera image embedding. Each embedding is a 768-vector. Image embedding is computed from a pretrained image encoder.

TABLE II
COSINE SIMILARITY WITH GROUND TRUTH CAMERA IMAGE

	10 deg	20 deg	30 deg	40 deg
LIDAR projection	0.20	0.33	0.43	0.43
Simple image translation	0.12	0.32	0.44	0.55

count how many LIDAR points are successfully mapped from the image pixels. A perfect TR would result in the maximum number of LIDAR points In practical, we use front object pixels from $\hat{I}$. The front object can be easily segmented using pre-trained models such as Segment Anything [15]. A problem with simple counting of the back-projected LIDAR points is that, for front-view objects, almost all pixels can be back-projected to a LIDAR point. This is particularly problematic if two candidate Tr's are both very close to the actual TR. The remedy we use is to use a norm filter to reject the LIDAR points that have a sudden change in the surface norm. For front-view objects, a smooth norm field can be reasonable assumed so we believe that such a filter would be able to provide the discriminating power we need.

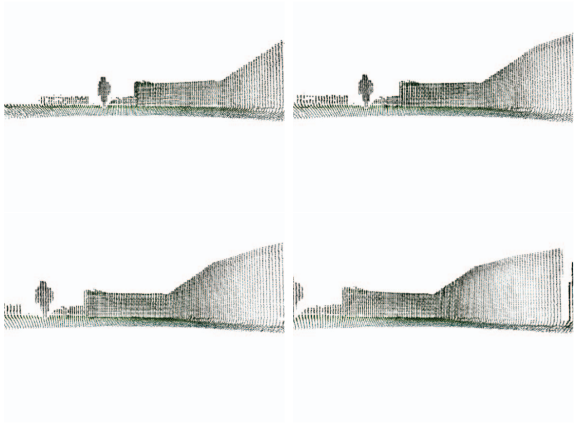

Fig. 3. Lidar projection example for four pose hyperthesis

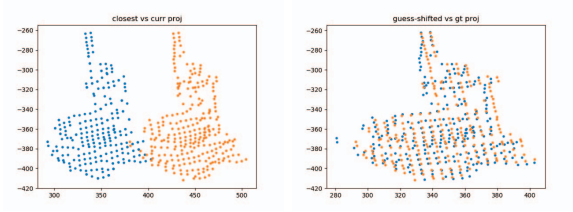

Fig. 4. LIDAR projection example for two pose hyperthesis, blue points are pixels extracted from camera image, orange points are LIDAR project of the same object : (left) initial projection, (right) corrected projection

TABLE III
SIMILARITY SCORE EXAMPLE USING EMBEDDING FROM THE TRAINED NETWORK. TESTING LIDAR PROJECTIONS ARE GENERATED WITH VARYING ROTATIONAL DITHERING.

	aligned	10 deg	20 deg	30 deg	40 deg
Similarity	1.0	0.710	0.6398	0.5717	0.43

IV. RESULTS AND DISCUSSION

Our experimental setup uses a Clearpath Husky robot to collect data. The Husky is equipped with an Ouster OS1 32-channel LIDAR sensor to collect the point cloud data. The image data is generated by an Intel Realsense D445 camera. Only the RGB camera data is used. The camera is mounted in front of the Ouster sensor. Our test scenario has the robot travel behind a human. Two additional human is further behind the robot. The field is a wooded area so the collected data contain both moving objects, ground plane, and static building objects. The distance of the Jackal is about 3 meters. We select LIDAR frames where the object points have considerable overlapping. We set the initial guess of the transformation to be 45 degrees off the ground truth in each of the roll, pitch, and yaw axes. The initial translation error is set to 0.5 meters in all axes. Figure 4 shows the result of our algorithm. The initial guess results in an unaligned projection as evidenced by the clearly separated object points. The figure on the right shows the result of the corrected transformation. The projected points are completely immersed with the camera image. Table III shows the error of the estimated transformation parameters corresponding to the simple closest matching and closest-affine matching. The mean translational errors for both are less than 5 center meters. The rotational error is measured at the three Euler angle axis. The mean rotation error is less than 2 degrees. Figure 4 shows the effect of camera-LIDAR calibration in 3D object detection pipeline based on 2D-3D diffusion [1]. With the uncalibrated camera-LIDAR parameter, about 50% of the object points are mislabeled. After calibration, the object points are labeled correctly.

Table III shows the performance of the trained network. We are interested in the quality of the generated embedding. As discussed earlier, the desired behavior is that the network will generate similar embedding if the LIDAR projection and the camera image are in sync. We indeed observed such a property: With the ground truth TR, the similarity score between the LIDAR projection and the camera image is 100%. As we introduce varying degrees of rotational error to TR, the similarity score gradually declines. For a 40-degree rotational error, the network predicts a 40% similarity score. These results shows strong evidence that the trained network has learned the measure similarity of the two different modalities.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we presented our preliminary study of utilizing a deep neural network to compare LIDAR projection and camera image, with a goal to generate a coarse estimation of the LIDAR-camera transformation. Our approach is novel as we use a contrastive network design to learn data representation even though their modalities are very different. Our results indicate that such a heterogeneous embedding comparison is feasible and even probably to achieve a general understanding of scene data.

In the future, we will explore the extension of the method under relaxed and sub-optimal conditions. One case is to handle situations when there are multiple foreground objects in the camera view. The current technique of object point mapping would need to be further validated.

ACKNOWLEDGEMENTS

The research was supported by U.S. Army grant #W911NF220016 and the AI for Maneuver and Mobility (AIMM) Essential Research Program (ERP) at DEVCOM Army Research Laboratory.

REFERENCES

[1] B. H. Wang, W.-L. Chao, Y. Wang, B. Hariharan, K. Q. Weinberger, and M. Campbell, "Ldls: 3-d object segmentation through label diffusion from 2-d images," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2902–2909, 2019.

[2] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," 2021.

[3] S. Eyuboglu, M. Varma, K. Saab, J.-B. Delbrouck, C. Lee-Messer, J. Dunnmon, J. Zou, and C. Ré, "Domino: Discovering systematic errors with cross-modal embeddings," 2022.

[4] B. Horn, "Fitting parameterized three-dimensional models to images," in *J. Optical Soc. Am.*, vol. 5, no. 7, 1987, pp. 1127–1135.

[5] D. Lowe, "Fitting parameterized three-dimensional models to images," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 13, no. 5, pp. 441–450, 1991.

[6] L. Quan and Z. Lan, "Linear n-point camera pose determination," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 21, no. 8, pp. 774–781, 1999.

[7] P. Besl and N. McKay., "A method for registration of 3-d shapes," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[8] A. V. Segal, D. Haehnel, and S. Thrun, "Generalized-icp," June 2009.

[9] J. Wang, V. R. Dasari, B. Geerhart, B. Rapp, and P. Wang, "Real-time camera-to-lidar calibration for autonomous robotic systems at the edge," in *2023 IEEE International Conference on Big Data (BigData)*. Los Alamitos, CA, USA: IEEE Computer Society, dec 2023, pp. 3868–3873.

[10] V. R. Dasari, M. S. Im, and B. Geerhart, "Complexity and mission computability of adaptive computing systems," *The Journal of Defense Modeling and Simulation*, vol. 19, no. 1, pp. 5–11, 2022.

[11] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," 2016.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[13] D. Cattaneo, M. Vaghi, and A. Valada, "Lcdnet: Deep loop closure detection and point cloud registration for lidar slam," 2022.

[14] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021.
[15] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.