Data-Efficient Decentralized Visual SLAM

Aishwarya Unnikrishnan[a], Lu Wen[b], Devesha Tewari[c], and Haonan Chang[d]

[a]University of Michigan, Robotics, shwarya@umich.edu
[b]University of Michigan, Mechanical Engineering, lulwen@umich.edu
[c]University of Michigan, Electric and Computational Engineering, devesha@umich.edu
[d]University of Michigan, Mechanical Engineering, harveych@umich.edu,

***Abstract*— This paper presents a data-efficient method for decentralized visual SLAM. This relies on cheap, lightweight camera sensors, and operates in a decentralized manner so it avoids reliance on a single central entity like in centralized systems. Typical decentralized systems exchange the entire map data between all robots. However, this results in very large data transfers, and a complexity scaling quadratically with the robot count. To reduce this, this method utilizes a compact image descriptor leveraging a state-of-the art visual vocabulary method and only sends data for relative pose estimation for a single matched robot. This reduces the complexity, scaling linearly with robot count. Decentralized optimization also uses a state-of-the-art decentralized pose-graph optimization method using a two-stage distributed Gauss-Seidel approach. Data-exchange is also minimized, and is linear with the trajectory overlap set. This method was evaluated on the KITTI dataset, and was tested for both data-efficiency and accuracy. We have made the following publicly available: our code[1] and our website [2].**

I. INTRODUCTION

Multi-robot systems surpasses single robots in many robotic applications. They are able to accomplish tasks more efficiently and quickly, and even execute tasks that cannot be done by a single robot. They also provide redundancy which ensures task completion, they increase flexibility of task execution and leverage distributed sensing and actuation. They are especially popular in exploration, navigation applications. For efficient collaboration among the agents in an environment lacking an absolute positioning system, multi-robot simultaneous localization and mapping (SLAM) is essential. That is, each robot must estimate the map (landmarks) and the state of all robots.

Visual SLAM has become popular due to the simple, cheap, lightweight and versatile camera sensor. [1] has presented numerous surveyed vSLAM algorithms. VSLAM has a higher technical difficulty than other sensor-based SLAMs since cameras can acquire less visual input from a limited field of views compared to 360°laser sensing. Thus, camera poses must be continuously estimated while simultaneously reconstructing the 3D structure of the unknown environment. Visual SLAM systems comprise 1) a visual odometry algorithm for initial state estimate, 2) a place recognition system for relating the currently observed scene to previously observed scenes 3) an optimization backend which consistently integrates the place matches from the place recognition system with the full stream of state estimates. The final end product is a map, which feeds back to the place recognition and optimization subsystems.

Multi-robot Visual SLAM can be done in a centralized or decentralized manner. In the former, there is a central control agent (robot or computer) that collects all the global information about the environment and the robots, and can communicate this with the robots. This allows it to simply produce globally optimal plans. However, it is ineffective for large robot teams (due to large computation and bandwidth requirement), not robust in relation to dynamic environments and failures in communications and causes a highly vulnerable system. On the other hand, decentralized systems do not suffer these consequences, but is more difficult to implement.

For this decentralized multi-robot visual SLAM system, visual odometry is run on each robot independently, converting sensor measurements into a pose estimate, and its output is fed to the rest of the decentralized SLAM system. Visual odometry algorithms also tend to exhibit drift, whereby metric accuracy decreases over time. Thus, if a robot returns to a previously visited location, the current pose estimate will most likely be inconsistent with the previous pose estimate. This is reduced by Place Recognition module, which recognizes these previously visited places despite the inconsistent poses. The optimization makes a consistent map.

This work combines state-of-the-art decentralized visual place recognition and decentralized optimization with a method for visual feature association. All subsystems are data-efficient, and their combination produces a data-efficient fully decentralized visual SLAM system.

II. METHODOLOGY

This method involves n robots, each executing the system shown in Fig.1.

Camera images are inputted into the visual odometry module and NetVLAD module [2]. The former produces intra-robot measurements for a sparse feature map for localization, while the latter produces a compact image representation for the decentralized visual place recognition (DVPR). The relative pose estimation module (RelPose) accepts the matches from the DVPR and the map from VO to establish relative poses between robot trajectories based on the VO sparse

[1]https://github.com/decentr-vslam/decentr-vslam
[2]https://decentr-vslam.github.io/Team13_Decentralized-Visual-SLAM

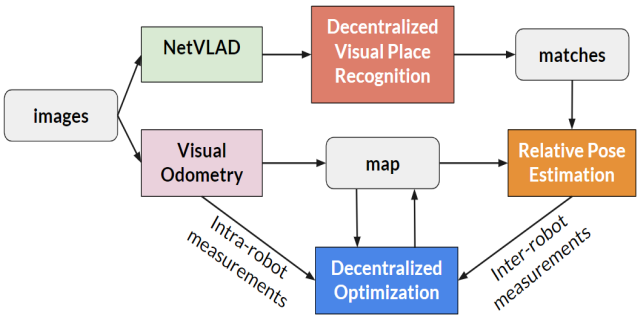

Fig. 1. System Overview

map and the candidate matches from NetVLAD. Finally, the decentralized optimization module (DOpt) continuously updates the map as new images are acquired. This is done based on the intra-robot relative pose measurements from the VO and the inter-robot relative pose measurements from RelPose module.

A. Visual Odometry Processing

The Visual Odometry (VO) algorithm accepts the camera images and produces intra-robot measurements. This includes the ORB descriptors, landmark positions, landmarks observations, poses, and times. This is required for the sparse feature map, which is used for the relative position estimation module. These intra-robot measurements are also used in optimization.

Any VO algorithm could be used once it satisfies the following requirements: 1) it produces a pose graph, 2) each pose is associated to an image to be used for place recognition, 3) it provides absolute scale. In this work, we implement visual odometry ORB-SLAM [3] in stereo configuration. Stereo configuration overcomes the following issues issues faced by monocular configuration: 1)lack of depth, 2) requirement of filtering techniques for an initial map, 3) scale drift.

B. Decentralized Visual Place Recognition

Decentralized visual place recognition (DVPR), then, becomes a key component of a decentralized visual SLAM system. Achieving it by having all robots send queries to all other robots would use vast amounts of bandwidth, and diverse approaches have been explored by the robotics community to reduce that bandwidth.

In this work, we instead implement a decentralized visual place recognition method that is based on full-image descriptors. The method consists in clustering the full-image descriptor space into several clusters and assigning each cluster to one robot [4]. As a result, place recognition can be achieved by sending each place query to only one robot.

1) Full-image descriptor: The whole CNN architecture for retrieving image descriptor is based on i) extracting local descriptors, which then is ii) pooled in an orderless manner. This is a widely followed pipeline in image retrieval as it can provide with lighting and viewpoint-changes-robust translation and partial occlusion. For step (i), the CNN is cropped at the last convolutional layer and viewed as a dense descriptor extractor. For step (ii) we leverage a new pooling layer, NetVLAD [2], that pools extracted descriptors into a fixed image representation.

Previous methods for image descriptor extraction is either incapable to extract global information from image, or too complex and costly for communication. Here we adopt a full image descriptor method, NetVLAD, as a basis, which is much less complex and needs to be sent to a single robot for each query.

NetVLAD is a new generalized VLAD layer, inspired by the "Vector of Locally Aggregated Descriptors" image representation commonly used in instance level retrieval and image classification. It captures information about the statistics of local descriptors aggregated over the image. The structure of the neural network used here is shown in fig 2. The CNN are composed of standard CNN layers, including convolutions, normalization, softmax layers and so on. The NetVLAD layer plays the role of an aggregation layer, joined up in a directed acyclic graph.

Given N D-dimensional local image descriptors $\{x_i\}$ as input, and K cluster centers $\{c_k\}$ as VLAD parameters, the output VLAD image representation is $K \times D$-dimensional. For convenience V is written as a $K \times D$ matrix. Intuitively, each $D-$dimensional column k of V records the sum of residuals $(x_i - c_k)$ of descriptors which are assign to cluster c_k. This matrix is intra-column-wise-normalized, converted into a vector, and finally L2-normalized in its entirety. Based on VLAD, Relja et.al constructed a layer amenable to training via backpropagation, which requires that the layer's operation is differentiable with respect to all its parameters and input. The (j,k) element of V is computed as follows:

$$V(j,k) = \sum_{i=1}^{N} a_k(\mathbf{x}_i)(x_i(j) - c_k(j)) \tag{1}$$

$$a_k(\mathbf{x}_i) = \frac{e^{-\alpha||x_i - c_k||^2}}{\sum_{k'} e^{-\alpha||x_i - c_{k'}||^2}} \tag{2}$$

where $a_k(\mathbf{x}_i)$ assigns the weight of descriptor $\mathbf{x}_i$ to cluster c_k proportional to their proximity, but relative to proximities to other cluster centers. "Training procedure and database:" The NetVLAD network is trained with Google Street View Time Machine database, which contains large amounts of weakly labelled imagery depicting the same places over time. For the place recognition task, the loss is designed as a weakly supervised triplet ranking loss that can deal with incomplete and noisy position annotations. We aim to get that the Euclidean distance $d_\theta(q, I_i)$ between the query q and a close-by image I_{i_*} to be smaller than the distance to far away images in the database. From the Google Street View Time Machine data, we obtain a training dataset of tuples $(q, \{p_i^p\}, \{n_j^q\})$, where for each training query image q we have a set of potential positives $\{p_i^q\}$ and the set of definite negatives $\{n_i^q\}$. Based on the intuition that, distance between the query q and the best matching potential postive p_{i*}^q is smaller than the distance $d_\theta(q, n_j^q)$ between the query

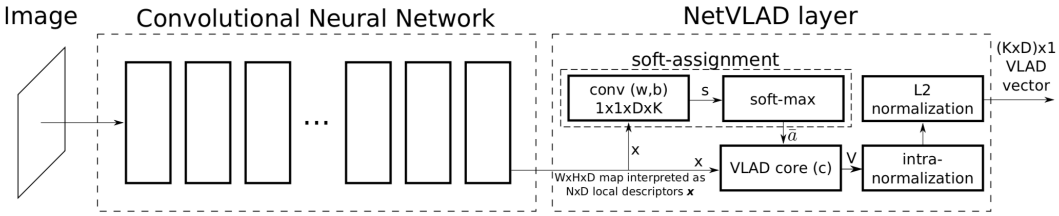

Fig. 2. CNN structure with NetVLAD later

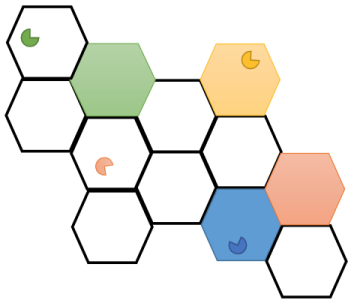

Fig. 3. Clustering and Voronoi cell
NetVLAD space is clustered based on the $k-$means of NetVLAD, and each center is assigned to robots. The color of cells and robots show the pre-assigning results. For each query, it's only sent to the robot which is pre-assigned to the cluster, which is identified as the voronoi cell that the query image belongs to.

q and all negative images q_j, the weakly supervised ranking loss L_θ for a training tuple is defined as:

$$L_\theta = \sum_j l(\min_i d_\theta^2(q, p_i^q) + m - d_\theta^2(q, n_j^q)) \tag{3}$$

where l is the hinge loss $l(x) = \max(x, 0)$, and m is a constant parameter giving the margin.

Compared with other methods, it is readily pluggable into any CNN architecture and amenable to training via backpropagation.

2) Place Matching: Place recognition aims to determine whether there is any image I_{β_j} of the same scene previously captured by another robot, given its latest image I_α. In this method, we firstly cluster the NetVLAD descriptor space into several clusters. Each robot γ is pre-assigned to a cluster center c_γ and the query v_α is sent only to robot $\delta = \arg\min_\gamma ||v_{\alpha_i} - c_\gamma||_{l_2}$, which is pre-assigned to the cluster that has the least l_2 distance between the query image. Place matching will be taken in the Voronoi cell:

$$V_\delta = \{v | c_\delta = \arg\min_{c_\gamma} ||v - c_\gamma||_{l_2}\} \tag{4}$$

Then the robot δ replies with the identifier of I_{β_j}:

$$(\beta, j) = \arg \min_{(\gamma,k):v_{\gamma_k} \in V_\delta} ||v_{\alpha_i} - v_{\gamma_k}||_{l_2} \tag{5}$$

where β, j stands for the optimal robot among all robots, and the optimal frame among all previous frames of the corresponding robot $]beta$ respectively.

C. Relative Pose Estimation

After place matching is executed, and there is a matched robot β at frame j, i.e. $(\beta,\ j)$, then robot α sends a relative pose estimation request to robot β. As a data-efficient approach, the only data sent from robot α to robot β is a set of visual word identifiers (w_k) and 3D positions of the landmarks (p_k), both corresponding to the keypoint k expressed in the camera frame of I_{β_j}. This is $d_{\text{RelPose}}^{\alpha\to\beta}$ which is defined in Eq.6

$$d_{\text{RelPose}}^{\alpha\to\beta} = (\{(w_k, p_k) \forall k \in K_{\alpha_i}\}, \alpha, i, j) \tag{6}$$

These landmarks are from the VO, and thus the ORB-SLAM implemented. However, ORB-SLAM [3] uses data surrounding the matches, which leads to expensive data exchange in a decentralized system. Thus, α to β, $d_{\text{RelPose}}^{\alpha\to\beta}$ uses the visual words w_k. [5] presents a visual data association method to achieve this by associating these high-dimensional descriptors (ORB) to a word index. These are then used for matching keypoints between images as opposed to using the full descriptors, as there will be similar descriptors assigned to the same word.

Next, robot β uses $d_{\text{RelPose}}^{\alpha\to\beta}$ to establish pairs of matching keypoints:

$$\{(k, k') | w_k = w_{k'},\ \nexists\ l \in K_{\alpha_i} : w_l = w_k,\ \nexists\ l' \in K_{\beta_j} : w_{l'} = w_{k'}\} \tag{7}$$

Then, RANSAC is used with the corresponding pairs of landmark positions $\{(p_k, p_{k'})\}$ of these matching keypoints to determine an initial estimate of the inter-robot measurement, $\bar{\boldsymbol{z}}_{\beta_j}^{\alpha_i}$ and the corresponding inlier matches, $M = \{(k, k')^*\}$. This place match M is compared to a inliers threshold and is rejected if $|M| < \tau_{\text{inliers}}$. Else, the corresponding inlier landmark position pairs are accepted and used to get $\bar{\boldsymbol{z}}_{\beta_j}^{\alpha_i}$ that minimizes a robust 3D registration error shown in Eq.8.

$$\sum_{\{(p_k, p_{k'})^*\}} \rho_{\tau_{\text{loss}}}(\|p_k - (\bar{R} \cdot p_{k'} + \bar{t})\|_{l_2}^2) \tag{8}$$

The weight of the remaining keypoint match outliers are reduced by a robust weight loss function: $\rho_{\tau_{\text{loss}}}(s) = \arctan \frac{s}{\tau_{\text{loss}}}$ The relative pose estimation module also includes a consistency check to avoid relative pose outliers. The following lists the rules for accepting a relative pose $\bar{\boldsymbol{z}}_{\beta_j}^{\alpha_i}$:

1) There is another already accepted relative pose $\bar{\boldsymbol{z}}_{\beta_j}^{\alpha_{i'}}$ which is consistent with the pose and the pairs (α, i) and (α, i') are within a distance of τ_{cdist}.

2) There is no accepted relative pose $\bar{z}_{\beta_{j'}}^{\alpha_{i'}}$ but there is a previous pose which fulfils the above two conditions.

In order to further reduce the amount of data of this module, the geometric verification is constrained based on a mininimum distance specified by τ_{mdg}. This essentially skips matches which are close to already established relative poses. That is, α only sends $d_{\text{RelPose}}^{\alpha\to\beta}$ for geometric verification to β if there is no other frame of α matched to a frame of β such that $(\|t_{\alpha_i} - t_{\alpha_{i'}}\|_{l_2}) < \tau_{\text{mdg}}$. However, while this increases data-efficiency, this now assumes that the VO and RelPose modules are consistent enough so that this resulting subset of inter-robot measurements suffices to produce a globally-consistent state estimate.

D. Decentralized Optimization

Decentralized optimization in this project is a two-stage distributed Gauss-Seidal Approach. [6]

1) Rotation initialization via relaxation and projection: In the first stage, an estimation is solved from the following subproblem:

$$min_{R_{\alpha_i}\in SO(3),\forall\alpha\in\Omega,\forall i} \sum_{(\alpha_i,\beta_j)\in\epsilon} \omega_R^2 ||R_{\beta_j} - R_{\alpha_i}\hat{R}_{\beta_j}^{\alpha_i}||_F^2 \quad (9)$$

This object considers the relative rotation measurements between different robot. Since the Eq. 9 has non-convex constraints, some techniques should be used to make it convex. In this work, approximation and projection technique is used.

Firstly, the optimization problem can be transformed into a quadratic problem by using convex relaxation.

$$min_{R_{\alpha_i}\forall\alpha\in\Omega,\forall i} \sum_{(\alpha_i,\beta_j)\in\epsilon} \omega_R^2 ||R_{\beta_j} - R_{\alpha_i}\hat{R}_{\beta_j}^{\alpha_i}||_F^2 \quad (10)$$

Here the $SO(3)$ constraints are removed. Because The Eq. 10 is actually a quadratic problem over R_{α_i} can be further written as:

$$min_r ||A_r r - b_r||^2 \quad (11)$$

This is actually a least square regression. Thus it can be directly solved by normal equations.

$$(A_r^T A_r) r = A_r^T b_r \quad (12)$$

However, the solution of this least square regression, which is denoted as $\hat{r}$ is actually not a rotation matrix. So an additional projection step should be further added. Denote $\tilde{R}_{\alpha_i}$ to be matrix form of $\hat{r}$. Then

$$\hat{R_{\alpha_i}} = project(\tilde{R_{\alpha_i}}) \quad (13)$$

The projection is done through SVD, which is described in this work.

2) Full pose recover via single GN iteration: In step 1, an estimation is got from convex relaxation and projection. However such an estimation may not be optimal. To modify it, we write the true rotation R_{α_i} as the estimation $\hat{R_{\alpha_i}}$ with an perturbation θ_{α_i}.

$$R_{\alpha_i} = \hat{R}_{\alpha_i} Exp(\theta_{\alpha_i}) \quad (14)$$

Then by taking R_{α_i} into Eq. 9, we can get the following objective function:

$$\begin{aligned} min_{t_{\alpha_i},\theta_{\alpha_i}\in R^3,\forall\alpha\in\Omega,\forall i} & \sum_{(\alpha_i,\beta_j)\in\epsilon} w_t^2 ||t_{\beta_j} - t_{\alpha_i} - \hat{R}_{\alpha_i} Exp(\theta_{\alpha_i})\bar{t}_{\beta_j}^{\alpha_i}||^2 \\ & + \frac{\omega_R^2}{2} ||\hat{R}_{\beta_j} Exp(\theta_{\beta_j}) - \hat{R}_{\alpha_i} Exp(\theta_{\alpha_i})\bar{R}_{\beta_j}^{\alpha_i}||_F^2 \end{aligned} \quad (15)$$

The formulation of this problem enable us to remove the constraints of $SO(3)$. However the non-convexity still exists in Exp. To solve this problem, an approximation can be made through a first order approximation on Exp.

$$Exp(\theta_{\alpha_i}) = I_3 + S(\theta_{\alpha_i}) \quad (16)$$

Here $S(\theta_{\alpha_i})$ is a skew symmetric matrix defined by vector θ_{α_i}. Taking it into previous formulation, we can get a new objective function.

$$\begin{aligned} min_{t_{\alpha_i},\theta_{\alpha_i}\in R^3,\forall\alpha\in\Omega,\forall i} & \sum_{(\alpha_i,\beta_j)\in\epsilon} \omega_t^2 ||t_{\beta_j} - t_{\alpha_i} - \hat{R}_{\alpha_i} S(\theta_{\alpha_i})\bar{t}_{\beta_j}^{\alpha_i}||^2 \\ & + \frac{\omega_R^2}{2} ||\hat{R}_{\beta_j} - \hat{R}_{\alpha_i}\bar{R}_{\beta_j}^{\alpha_i} + \hat{R}_{\beta_j} S(\theta_{\beta_j}) - \hat{R}_{\alpha_i} S(\theta_{\alpha_i})\bar{R}_{\beta_j}^{\alpha_i}||_F^2 \end{aligned} \quad (17)$$

Then by combining variables, we can reformulate this problem to a linear-squares problem.

$$min_p ||A_p p - b_p||^2 \quad (18)$$

The solution of this problem can be gotten through normal equation.

$$(A_p^T A_p) p = A_p^T b_p \quad (19)$$

Here p is a combination of position t_{α_i} and rotation correction θ_{α_i}. Finally the rotation matrix can be written as $R_{\alpha_i} = \hat{R}_{\alpha_i} Exp(\theta_{\alpha_i})$.

3) Distributed Trajectory Estimation: One important observation for the two-stage optimization method mentioned is that this optimization can be done in a decentralized way. Notice the final version of the two-stage optimization is least square regression. And least square regression problem is known that it can be done in a decentralized way through a distributed linear solver.

The first step of solution is to partition the unknown vectors into subvectors for different robot. For example,

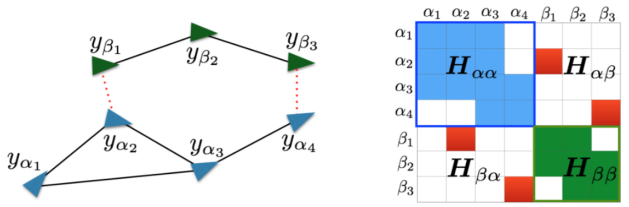

Fig. 4. Case of information exchange

vector r for the first stage, and vector p for the second stage can be written as:

$$r = [r_\alpha, r_\beta, \ldots], p = [p_\alpha, p_\beta, \ldots] \tag{20}$$

Since the two stage have a similar formulation, we can write them into an uniform form:

$$\begin{bmatrix} H_{\alpha\alpha} & H_{\alpha\beta} & \ldots \\ H_{\beta\alpha} & H_{\beta\beta} & \ldots \\ \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} y_\alpha \\ y_\beta \\ \vdots \end{bmatrix} = \begin{bmatrix} g_\alpha \\ g_\beta \\ \vdots \end{bmatrix} \tag{21}$$

Here is the problem formulation: we need to compute y given H and g. Once again, we need to partition the square matrix H and the vector g by the block structure of y.

We can get the update formula through a set of algebraic computation.

$$\sum_{\delta \in \Omega} H_{\alpha\delta} y_\delta = g_\alpha, \ \forall \alpha \in \Omega \tag{22}$$

Extract y_α from the sum:

$$H_{\alpha\alpha} y_\alpha = - \sum_{\delta \in \Omega / \ /\alpha/} H_{\alpha\delta} y_\delta + g_\alpha, \forall \alpha \in \Omega \tag{23}$$

This formula can be written as a iterative update formulation:

$$y_\alpha^{k+1} = H_{\alpha\alpha}^{-1} (- \sum_{\delta \in \Omega_\alpha^+} H_{\alpha\delta} y_\delta^{k+1} - \sum_{\delta \in \Omega_\alpha^-} H_{\alpha\delta} y_\delta^{(k)} + g_\alpha \tag{24}$$

The update starts from an arbitrary initial estimation $y^{(0)} = [y_\alpha^{(0)}, y_\beta^{(0)} \ldots]$. Ω_α^+ is the set robots which has been updated in current time step, and Ω_α^- is the set of robots which has not been updated in current time step. What these iterative update functions do is that in each iteration, they update the current robot state with the latest state from the other robots. In this way, the two steps of Gaussian Seidal optimization can be solved decentralized.

4) Information exchange in distributed GS optimization: From the update formula, we know the formulation of decentralized optimization. Iterative optimization is done separately on different robot. The only information transition needed between different robots are $H_{\alpha\beta}$ and latest robot state.

Here is a simple example for illustration in fig. 4, two robots are in this map separately. According to the data association, there are only two non-zero values in $H_{\alpha\beta}$. Furthermore, because there are only two robot frame paired with each other, the state information transition can be limited into 2 states, which is very data efficient.

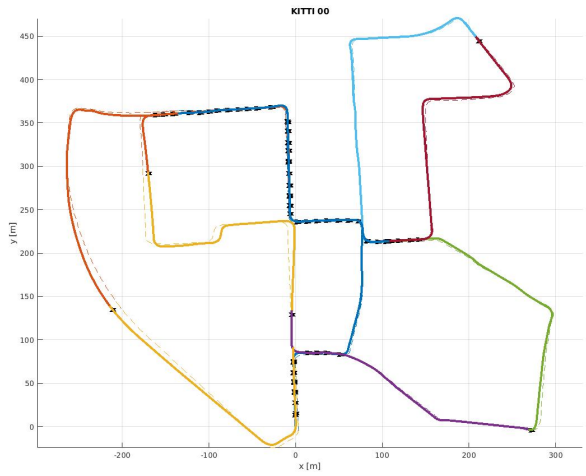

Fig. 5. Final plot using 10 robots on KITTI 00

5) Convergence of decentralized optimization: The Gauss-Seidel optimization has been proved to converge with an arbitrary initial value, which is very important for a distributed system.

III. Experiments and Results

In this section, we will d the implementation of our decentralized visual SLAM. Results and analysis are included to evaluate our approach. All plots shown in this section are plotted after the simulation is complete. 5 shows the overall final plot using 10 robots on KITTI 00.

A. Experiments

We build the CNN with a NetVLAD layer structure with TensorFlow, and test the results with that provided by [2] on 24/7 Tokyo dataset [7]. We compare our training network with that given by [8], and the average deviation between our NetVLAD results and that of theirs on KITTI dataset is 0.157 (after normalization).

Fig. 6. 24/7 Tokyo dataset

We evaluated our system using the KITTI dataset [9], which contains stereo sequences recorded from a car in urban and highway environments. We use the grey-scale images of sequence 00 as well as the raw odometry data. The data is then divided into n parts to simulate trajectories recorded by n robots. The original trajectory as well as the divided trajectory to simulate multiple robots are shown in Fig.7.

B. Results

1) Information Exchange Analysis: Since the project is focused on data efficient decentralized optimization, we

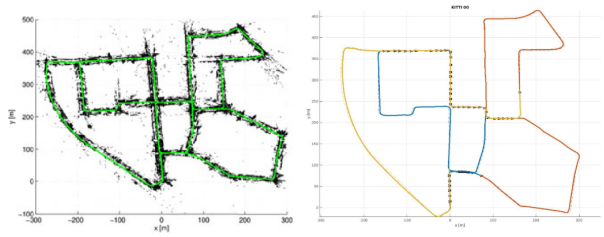

Fig. 7. Split for multi-robot on kitti dataset
Left: Original Kitti Data. Right: Kitti Data split for multi-robots (shown in different colors)

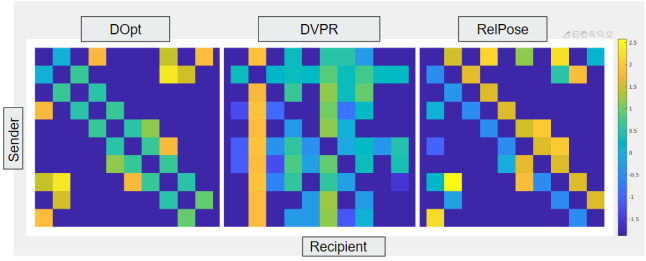

Fig. 8. Plot for data exchange of each of the 10 robots with each other over the course of the simulation during the three main stages of the algorithm. Left: DOpt, Center: DVPR, Right: RelPose

analyse the data exchange between robots. Thus, a heatmap that demonstrates the data exchange (on a log map) is plotted in Fig. 8. The sparser the heatmap (dark blue), the less the data exchanged. It is evident that in the DVPR step accounts for the most attempts at data exchange. This is because the place recognition attempts to query the frames of the other robots every time it is at the cluster center of another robot. Only if the verification server accepts the match between the frames, the relative pose is computed, after which, the optimization may happen.

As we already explained in Figure 7, successive robots have overlaps i.e. robot 1 with robot 2 and robot 2 with 3 and so on... This explains why for the DOpt and RelPose heatmap, the bytes transmitted along the diagonal are higher. Therefore, the loop closures in the simulation are effectively found and updates in DOpt and RelPose happen accordingly.7

In Figure 9, the total data transmitted One thing worth noting is that after a total process of 100s, the total data transmission is still less than 1.2 MB, which is fairly remarkable.

C. Accuracy Analysis

Fig. 10 is showing the average trajectory error of each robots in blue and the connected frame of each robot with others in orange. From this figure, we can find that the trajectory error keeps increasing at the first 50 seconds. However, as the connected frames become more and more, the trajectory error gets a huge decrease, and reaches a stable level of about of 5 meters, which is relative small compared with error peak which is about 30 meters.

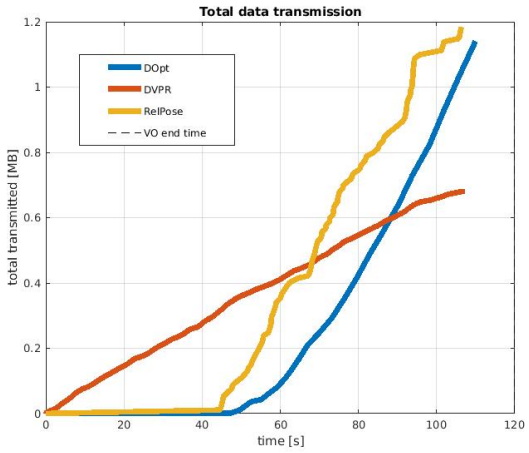

Fig. 9. Total amount of data transmission. Notice how DOpt takes the byte transmission because sending the map is an expensive process but querying frames using DVPR despite happening more frequently transmits less bytes bytes of information overall.

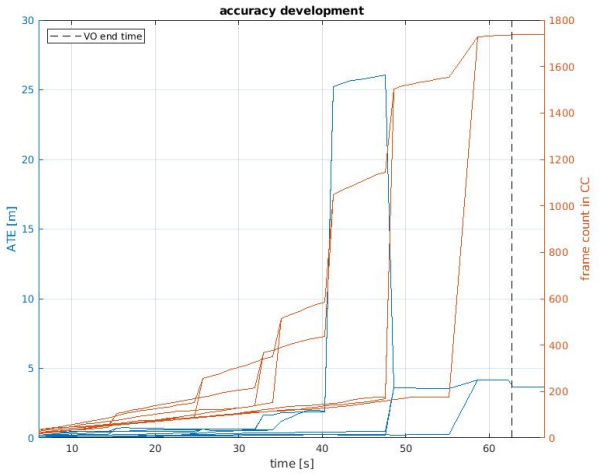

Fig. 10. Accuracy of SLAM process and connected frames for each robot. Blue lines are trajectory estimation error for different robots and orange lines are their connected frames by the time step.

IV. CONCLUSION

We have implemented a decentralized visual SLAM system, optimizing different modules of the process such as Decentralized Visual Place Recognition, Relative Pose Estimation and Decentralized Optimization to maximize data-efficiency. Overall, this system minimizes the data transmitted by each robot for decentralized SLAM by determining which robots have the relevant information (matches) and limiting communication with those. The system has been evaluated for accuracy and data-efficiency using a sequence of the KITTI dataset. This showed that this data-efficient algorithm can be adopted without affecting the accuracy of the system. Finally, our implementation was significantly faster than the original MATLAB implementation.

V. FUTURE WORK

Firstly, with respect to visual odometry, this system associates each pose to an image for place recognition using a single camera-setup. It can be useful to expand to a multi-robot setup. Also, we can further implement this using ORB-SLAM with IMU data. Thirdly, this method assigned cluster centers initially, however, doing such dynamically during execution can be investigated further.

VI. ACKNOWLEDGEMENTS

This implementation was based on the paper "Data-Efficient Decentralized Visual SLAM". We would like to acknowledge the authors Titus Cieslewski, Siddharth Chourdhary and Davide Scaramuzza. Also, thank you to Maani Ghaffari Jadidi, our course instructor for his support.

REFERENCES

[1] T. Taketomi, H. Uchiyama, and S. Ikeda, "Visual slam algorithms: a survey from 2010 to 2016," *IPSJ Transactions on Computer Vision and Applications*, vol. 9, no. 1, p. 16, Jun 2017. [Online]. Available: https://doi.org/10.1186/s41074-017-0027-2

[2] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "Netvlad: Cnn architecture for weakly supervised place recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5297–5307.

[3] R. Mur-Artal and J. D. Tards, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, Oct 2017.

[4] T. Cieslewski and D. Scaramuzza, "Efficient decentralized visual place recognition from full-image descriptors," in *2017 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. IEEE, 2017, pp. 78–82.

[5] D. Tardioli, E. Montijano, and A. R. Mosteo, "Visual data association in narrow-bandwidth networks," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2015, pp. 2572–2577.

[6] C. N. J. R. H. I. C. F. D. Siddharth Choudhary, Luca Carlone, "Distributed trajectory estimation with privacy and communication constraints: a two-stage distributed gauss-seidel approach," *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[7] A. Torii, R. Arandjelović, J. Sivic, M. Okutomi, and T. Pajdla, "24/7 place recognition by view synthesis," in *CVPR*, 2015.

[8] T. Cieslewski, S. Choudhary, and D. Scaramuzza, "Data-efficient decentralized visual slam," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 2466–2473.

[9] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: the kitti dataset," *The International Journal of Robotics Research*, vol. 32, pp. 1231–1237, 09 2013.

[10] N. Atanasov, J. Le Ny, K. Daniilidis, and G. J. Pappas, "Decentralized active information acquisition: Theory and application to multi-robot slam," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 4775–4782.