Maps from Motion (MfM): Generating 2D Semantic Maps from Sparse Multi-view Images - Supplementary Material

Matteo Toso[1] Stefano Fiorini[1] Stuart James[1,2] Alessio Del Bue[1]

[1]Istituto Italiano di Tecnologia (IIT) {name.surname}@iit.it
[2]Durham University stuart.a.james@durham.ac.uk

1. Derivation of the 2D Alignment Algorithm

In Section 3.3 of the main paper, we introduce a *Self-Consistency Loss* that requires finding the optimal transformation aligning two maps, *i.e*. two sets of 2D points, in the same reference frame. To train the model, we need this loss to be differentiable, with stable gradients and efficient to compute.

To satisfy these requirements, we introduce a novel alignment algorithm constrained to a 2D surface and inspired by the Procrustes algorithm. Consider two 2D points c_i and c_j where $c_i, c_j \in \mathcal{R}^{N\times 2}$, with the same number of elements and sorted such that the n-th element of c_i and c_j represent the same object but in different reference frames. These 2D point arrangements can be aligned by finding the rigid transformation defined by a rotation angle θ, a scaling factor λ, and a translation τ that solves the problem:

$$\Theta(c_i, c_j) = \underset{\theta,\tau,\lambda}{\arg\min} \|c_i - (\lambda R(\theta)\cdot c_j + \tau)\|. \quad (1)$$

To find this transformation, we first center and normalize the two point clouds, to reduce the alignment problem to estimating a rotation around the cloud's centers:

$$\tau_i = \frac{1}{N}\sum c_i \qquad \tau_j = \frac{1}{N}\sum c_j \quad (2)$$

$$\lambda_i = \|c_i - \mu_i\| \qquad \lambda_j = \|c_j - \mu_j\| \quad (3)$$

$$c'_i = \frac{c_i - \tau_i}{\lambda_i} \qquad c'_j = \frac{c_j - \tau_j}{\lambda_j}. \quad (4)$$

We can then compare the angular distribution of points in c'_i and c'_j, to estimate the rotation angle θ_{ji} that best aligns them. The orientation of point l in the point cloud c'_i, θ^i_l, can be obtained as $\gamma^i_l = \|c^l_i\|^{-1} c^l_i = [\cos(\theta^i_l), \sin(\theta^i_l)]$, *i.e*. by normalizing the vector connecting the point to the center of the cloud μ_k. The two angular distributions are found as $\gamma_i = \left\{[\cos(\theta^i_l), \sin(\theta^i_l)]\right\}_{l\in c'_i}$ and $\gamma_j = \left\{[\cos(\theta^j_l), \sin(\theta^j_l)]\right\}_{l\in c'_j}$.

Due to the possible presence of noise in the input 2D point clouds, we compute the average angular distance between pairs of matched points to find the rotation angle θ_{ji} mapping the two distributions. This is performed using the Prosthaphaeresis formulas, which can be computed as follows:

$$\sin\theta_{ji} = \frac{1}{N}\sum_l (\sin\theta^l_j \cos\theta^l_i - \cos\theta^l_j \sin\theta^l_i) \quad (5)$$

$$\cos\theta_{ji} = \frac{1}{N}\sum_l (\cos\theta^l_j \cos\theta^l_i - \sin\theta^l_j \sin\theta^l_i) \quad (6)$$

$$R(\theta_{ji}) = \begin{bmatrix} \cos\theta_{ji} & \sin\theta_{ji} \\ -\sin\theta_{ji} & \cos\theta_{ji} \end{bmatrix}. \quad (7)$$

The rotation $R(\theta_{ji})$, the scaling factor λ_i and the translation τ_i then define the optimal transformation mapping c_j onto c_i, *i.e*. $c_i \approx \lambda_i R(\theta_{ji})\cdot c_j + \tau_i$.

2. Graph Formulation for the MfM Problem

To provide more insight in the process of encoding multiple local semantic maps into a graph structure, we highlight in Figure 1 the structure of the connectivity matrix, detailing how the objects are turned into nodes (1.a); how to draw same-map (1.b) and same-class (1.c) connections; and how to initialize the embeddings (1.d). The process is also summarized as pseudocode in Algorithm 1.

3. Ablation on Loss Components

In Section 3.3 of the main paper, we propose applying a combination of three loss functions (Euclidean Camera-Object Pose, Cross-Map Consistency, and Self-Similarity) to train the *GNN-based Alignment Module*. In this section, we highlight the effect of varying the combinations of loss functions, as our goal is to explore the advantages of utilizing these three losses simultaneously. Using MfM with

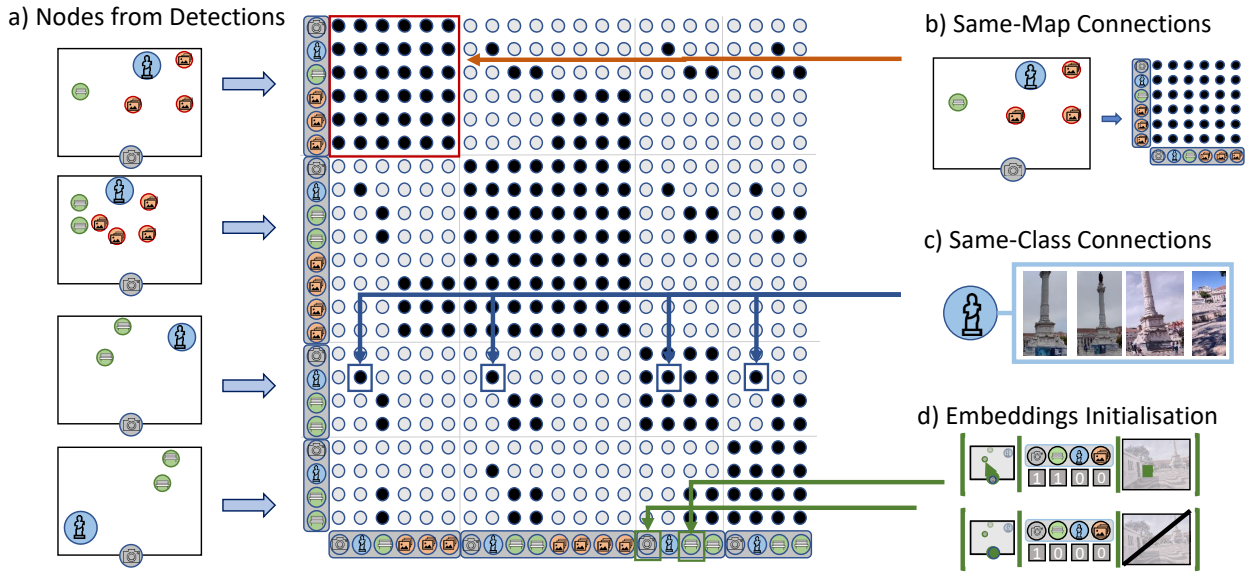

Figure 1. *Graph Formulation for the MfM problem.* Given a set of local maps, we *a)* assign to each map annotation a node in the graph and *b)* draw intra-map edges to generate a complete subgraph for each map. We then *c)* draw inter-map edges connecting detections matched to the same object. Finally, we *d)* assign to each node of the graph an embedding defined by concatenating the detection's coordinates in the corresponding local map, the one-hot encoding of its semantic class, and the location of a bounding box fitted to the segmentation mask.

Data: Images I_i, Objects O_i, Cameras C_i, Objects' Classes L_i
Result: Graph $\tilde{\mathcal{G}}$
`// Subgraph Definition`
for *each image* I_i **do**
  $V_i = O_i \cup C_i$
  $E_i = \{(u, v) \mid \forall u, v \in V_i\}$
  $G_i = \{V_i, E_i\}$
end
`// Large Graph Definition`
$L_i = \{l_{ij}\}_{j \in [1,\dots,|O_i|]}$
$\tilde{O} = \cup_{G_i} O_i$
$\tilde{V} = \cup_{G_i} V_i$
for *all objects* $o_s \in O_i$ **do**
  for *all object* $o_t \in O_m$ **do**
    $E_{st} = (v_s, v_t)$ if $l_{is} = l_{mt}$, where $v_s, v_t \in \tilde{V}$
  end
end
$E_l = \{E_{st}\}_{s \in [1,\dots,|\tilde{O}|], t \in [1,\dots,|\tilde{O}|]}$
$\tilde{E} = (\cup_{G_i} E_i) \cup E_l$
$\tilde{\mathcal{G}} = \{\tilde{V}, \tilde{E}\}$

Algorithm 1: Graph Formulation for the MfM problem.

a TransformerGCN network, we show results on both the MfM Dataset Small and Large scenes in the two opposite configurations: *i)* graph with no known matches between the detections in different views and noisy local maps as input (Depth Local Maps + Class-based Correspondences), and *ii)* graph with ground-truth detection matches and the ground truth local maps (GT Local Maps + GT Detection Matches).

As shown in Table 1, defining the loss function to incorporate all three losses proves to be advantageous for the overall predictive performance. Across both datasets and their various configurations, the MfM baseline, utilizing all three losses, consistently attains good results. Conversely, when the self-similarity loss is omitted, there is a noticeable decline in performance. The exclusion of either Euclidean Camera-Object Pose Loss or Cross-Map Consistency Loss leads to increased fluctuations in the results.

4. Ablation on Synthetic Dataset

In the main paper, we report results on synthetic scenes using 8 cameras observing a scene with 7 object of 5 possible classes. These parameters were decided on empirical bases, after testing different combination of object and classes sizes. In this section, we summarize the results of such hyperparameter tuning. We set the visibility parameter to 1.0 ($\phi = 1.0$) and the noise level on the objects' locations in the local maps to 0 ($\Delta_{xy} = 0$). We report the results based on the average Euclidean error on the reconstructed object locations (μ_o).

Depth Local Maps + Class-based Correspondences						
Method	Small Scenes			Large Scenes		
	Fail (%)	$\mu_c \pm \sigma_c$	$\mu_o \pm \sigma_o$	Fail (%)	$\mu_c \pm \sigma_c$	$\mu_o \pm \sigma_o$
MfM Baseline	37	3.7 ± 2.2	3.5 ± 1.6	10	3.8 ± 1.8	2.7 ± 1.6
MfM w/o Euclidean Cam-Obj Pose	36	3.7 ± 2.1	3.5 ± 1.5	10	3.9 ± 1.7	2.6 ± 1.4
MfM w/o Cross-Map Consistency	37	3.5 ± 2.2	3.5 ± 1.5	10	3.7 ± 1.7	3.0 ± 1.6
MfM w/o Self-Similarity	40	3.8 ± 2.2	4.2 ± 1.5	10	3.8 ± 1.7	3.1 ± 1.5
GT Local Maps + GT Detection Matches						
Method	Small Scenes			Large Scenes		
	Fail (%)	$\mu_c(m) \pm \sigma_c(m)$	$\mu_o(m) \pm \sigma_o(m)$	Fail (%)	$\mu_c(m) \pm \sigma_c(m)$	$\mu_o(m) \pm \sigma_o(m)$
MfM Baseline	31	3.9 ± 2.2	3.6 ± 1.4	5	3.7 ± 1.7	2.6 ± 1.2
MfM w/o Euclidean Cam-Obj Pose	35	3.7 ± 2.2	3.5 ± 1.6	5	3.8 ± 1.8	2.4 ± 1.1
MfM w/o Cross-Map Consistency	31	3.9 ± 2.2	3.6 ± 1.5	5	3.7 ± 1.9	3.1 ± 1.4
MfM w/o Self-Similarity	31.9	3.7 ± 2.1	4.1 ± 1.4	5	4.2 ± 2.0	3.1 ± 1.7

Table 1. Given a set of local 2D semantic maps from the Small and Large sequences of the MfM Dataset, we report the performance of MfM using *i)* noisy inputs (Depth Local Maps + Class-based Correspondences) and *ii)* perfect inputs (GT Local Maps + GT Detection Matches). Results include the average Euclidean error on the reconstructed object locations (μ_o), and its standard deviation (σ_0). We report the fraction of scenes for which the reconstruction failed (Fail).

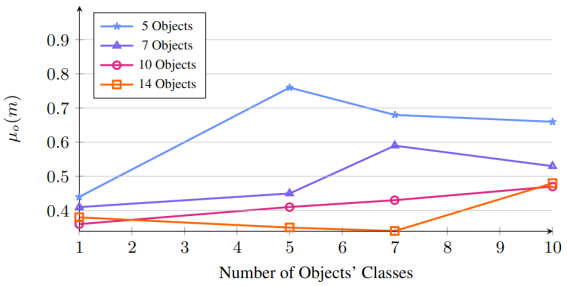

Figure 2. Average Euclidean error on the reconstructed object locations (μ_o).

The results, as illustrated in Figure 2, reveal the trend where when the number of objects increases, the reconstructed object location error decreases. This observation underscores the correlation between the number of objects and the increased accuracy in the process of object localization.

5. Experiment Details

Hardware. The experiments were conducted on a machine with an NVIDIA RTX 4090 GPU, 64 GB RAM, and 12th Gen Intel(R) Core(TM) i9-12900KF CPU @ 3.20GHz.

Model Setting. We train MfM in combination with different models using Adagrad as the optimization algorithm [1]. During our training process, we set a maximum of 1000 epochs, but we stopped the training earlier to prevent unnecessary iterations when the validation error no longer decreases. We optimized the weight decay using the *Bayesian optimization* technique and obtained two different results based on the dataset:

- **MfM Real Dataset.** We set the weight decay to 0.007.
- **MfM Synthetic Dataset.** We set the weight decay to 0.046.

References

[1] Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. Journal of machine learning research **12**(7) (2011) 3