Path-Invariant Map Networks

Zaiwei Zhang
UT Austin

Zhenxiao Liang
UT Austin

Lemeng Wu
UT Austin

Xiaowei Zhou
Zhejiang University

Qixing Huang
UT Austin

Abstract

Optimizing a network of maps among a collection of objects/domains (or map synchronization) is a central problem across computer vision and many other relevant fields. Compared to optimizing pairwise maps in isolation, the benefit of map synchronization is that there are natural constraints among a map network that can improve the quality of individual maps. While such self-supervision constraints are well-understood for undirected map networks (e.g., the cycle-consistency constraint), they are underexplored for directed map networks, which naturally arise when maps are given by parametric maps (e.g., a feed-forward neural network). In this paper, we study a natural self-supervision constraint for directed map networks called path-invariance, which enforces that composite maps along different paths between a fixed pair of source and target domains are identical. We introduce path-invariance bases for efficient encoding of the path-invariance constraint and present an algorithm that outputs a path-variance basis with polynomial time and space complexities. We demonstrate the effectiveness of our formulation on optimizing object correspondences, estimating dense image maps via neural networks, and 3D scene segmentation via map networks of diverse 3D representations. In particular, our approach only requires 8% labeled data from ScanNet to achieve the same performance as training a single 3D segmentation network with 30% to 100% labeled data.

1. Introduction

Optimizing a network of maps among a collection of objects/domains (or map synchronization) is a central problem across computer vision and many other relevant fields. Important applications include establishing consistent feature correspondences for multi-view structure-from-motion [1, 11, 44, 5], computing consistent relative camera poses for 3D reconstruction [20, 18], dense image flows [57, 56], image translation [59, 52], and optimizing consistent dense correspondences for co-segmentation [47, 16, 48] and object discovery [40, 8], just to name a few. The benefit of optimizing a map network versus optimizing maps between pairs of objects in isolation comes from the *cycle-consistency* constraint [31, 17, 15, 47], namely composite maps along cycles should be the identity map. For example, this constraint allows us to replace an incorrect map between a pair of dissimilar objects by composing maps along a path of similar objects [17]. Computationally, state-of-the-art map synchronization techniques [3, 7, 15, 19, 43, 16, 17, 25, 57, 58, 29] employ matrix representations of maps [25, 17, 15, 48, 16]. This allows us to utilize a low-rank formulation of the cycle-consistency constraint (c.f. [15]), leading to efficient and robust solutions [16, 58, 43, 19].

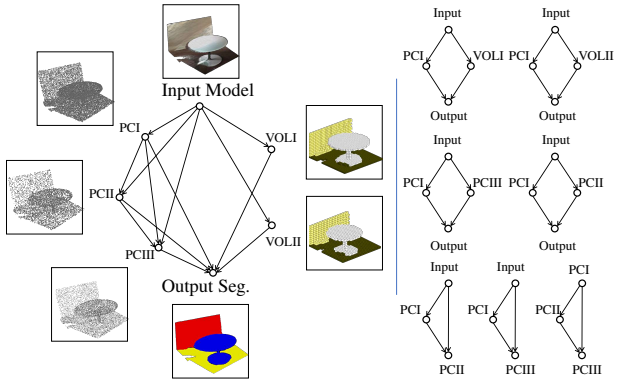


Figure 1: (Left) A network of 3D representations for the task of semantic segmentation of 3D scenes. (Right) Computed path-invariance basis for regularizing individual neural networks.

In this paper, we focus on a map synchronization setting, where matrix-based map encodings become too costly or even infeasible. Such instances include optimizing dense flows across many high-resolution images [30, 24, 41] or optimizing a network of neural networks, each of which maps one domain to another domain (e.g., 3D semantic segmentation [12] that maps the space of 3D scenes to the space of 3D segmentations). In this setting, maps are usually encoded as broadly defined parametric maps (e.g., feed-forward neural networks), and map optimization reduces to optimizing hyper-parameters and/or network parameters. Synchronizing parametric maps introduces many technical challenges. For example, unlike correspondences between objects, which are undirected, a parametric map may not have a meaningful inverse map (e.g., a neural network that takes a shape as input and outputs its semantic label). This raises the challenge of formulating an equivalent regularization constraint of cycle-consistency for directed map net-

works. In addition, as matrix-based map encodings are infeasible for parametric maps, another key challenge is how to efficiently enforce the regularization constraint for map synchronization.

We introduce a computational framework for optimizing directed map networks that addresses the challenges described above. Specifically, we propose the so-called *path-invariance constraint*, which ensures that whenever there exists a map from a source domain to a target domain (through map composition along a path), the map is unique. This path-invariance constraint not only warrants that a map network is well-defined, but more importantly it provides a natural regularization constraint for optimizing directed map networks. To effectively enforce this path-invariance constraint, we introduce the notion of a *path-invariance basis*, which collects independent path pairs that can induce the path-invariance property of the entire map network. We also present an algorithm for computing a path-invariance basis from an arbitrary directed map network. The algorithm possesses polynomial time and space complexities.

We demonstrate the effectiveness of our approach on three settings of map synchronization. The first setting considers undirected map networks that can be optimized using low-rank formulations. [16, 58]. Experimental results show that our new formulation leads to competitive and sometimes better results than state-of-the-art low-rank formulations. The second setting studies consistent dense image maps, where each pairwise map is given by a neural network. Experimental results show that our approach significantly outperforms state-of-the-art approaches for computing dense image correspondences. The third setting considers a map network that consists of 6 different 3D representations (e.g., point cloud and volumetric representations) for the task of semantic 3D semantic segmentation (See Figure 1). By enforcing the path-invariance of neural networks on unlabeled data, our approach only requires 8% labeled data from ScanNet [12] to achieve the same performance as training a single semantic segmentation network with 30% to 100% labeled data.

2. Related Works

Map synchronization. So far most map synchronization techniques [20, 18, 54, 31, 15, 57, 16, 7, 49, 5, 58, 2, 53, 19, 34, 43, 14, 56, 59, 52] have focused on undirected map graphs, where the natural regularization constraint is given by cycle-consistency. Depending on how the cycle-consistency constraint is applied, existing approaches fall into three categories. The first category of methods [20, 18] utilizes the fact that a collection of cycle-consistent maps can be generated from maps associated with a spanning tree. However, these approaches are only suitable for removing incorrect maps from the input maps, and it is hard to apply them for optimizing cycle-consistent neural networks, where the neural networks change during the course of the optimization. The second category of approaches [54, 31, 57] applies constrained optimization to select cycle-consistent maps. These approaches are typically formulated so that the objective functions encode the score of selected maps, and the constraints enforce the consistency of selected maps along cycles. The major advantage of these methods is that the correct maps are determined globally, leading to better performance than the first category of approaches. Our approach is relevant to this category of methods but addresses a different problem of optimizing maps along directed map networks. In particular, we introduce the path-invariance constraint and show how to enforce the path-invariance constraint effectively using path-invariance bases.

The third category of approaches apply modern numerical optimization techniques to optimize cycle-consistent maps. Along this line, people have introduced convex optimization [15, 16, 7, 49], non-convex optimization [5, 58, 2, 53, 19], and spectral techniques [34, 43]. To apply these techniques for parametric maps, we have to hand-craft an additional latent domain, as well as parametric maps between each input domain and this latent domain, which may suffer from the issue of sub-optimal network design. In fact, although people have applied such techniques for multilingual machine translation [21], existing approaches only work if the differences among the input domains are small and there exist meaningful bidirectional maps between them, leading to undirected map networks. In contrast, we focus on directed map networks among diverse domains and explicitly enforce the path-invariance constraint via path-invariance bases.

Joint learning of neural networks. Several recent works have studied the problem of enforcing cycle-consistency among a cycle of neural networks for improving the quality of individual networks along the cycle. Zhou et al. [56] studied how to train dense image correspondences between real image objects through two real-2-synthetic networks and ground-truth correspondences between synthetic images. [59, 52] enforce the bi-directional consistency of transformation networks between two image domains to improve the image translation results. However, in these works the cycles are explicitly given. In contrast, we study how to extend the cycle-consistency constraint on undirected graphs to the path-invariance constraint on directed graphs. In particular, we focus on how to compute a path-invariance basis for enforcing the path-invariance constraint efficiently. A recent work [55] studies how to build a network of representations for boosting individual tasks. However, self-supervision constraints such as cycle-consistency and path-invariance are not employed. Another distinction is that our approach seeks to leverage unlabeled data, while [55] focuses on transferring labeled data under different representations/tasks. Our approach is also related to model/data distillation (See [38] and the references therein), which can be considered as many edges between two domains. In

this paper, we focus on defining self-supervision for general graphs.

Cycle-bases of graphs. Path-invariance bases are related to cycle-bases on undirected graphs [22], in which any cycle of a graph is given by a linear combination of the cycles in a cycle-basis. However, besides fundamental cycle-bases [22] that can generalize to define cycle-consistency bases, it is an open problem whether other types of cycle-bases generalize or not. Moreover, there are fundamental differences between undirected and directed map networks. This calls for new tools for defining and computing path-invariance bases.

3. Path-Invariance of Directed Map Networks

In this section, we focus on the theoretical contribution of this paper, which introduces an algorithm for computing a path-invariance basis that enforces the path-invariance constraint of a directed map network. In Section 4, we show how to leverage this path-invariance basis to jointly optimize a directed map network to improve the maps in this network. Note that the proofs of theorems and propositions in this section are deferred to the Appendix.

3.1. Path-Invariance Constraint

We first define the notion of a directed map network:

Definition 1. *We define a directed map network $\mathcal{F}$ as an attributed directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $V = \{v_1, \ldots, v_{|\mathcal{V}|}\}$. Each vertex $v_i \in \mathcal{V}$ is associated with a domain $\mathcal{D}_i$. Each edge $e \in \mathcal{E}$ with $e = (i, j)$ is associated with a map $f_{ij} : \mathcal{D}_i \to \mathcal{D}_j$. In the following, we always assume $\mathcal{E}$ contains the self-loop at each vertex. The map associated with each self-loop is the identity map at the corresponding domain.*

For simplicity, whenever it can be inferred from the context we simplify the terminology of a directed map network as a map network. The following definition considers induced maps along paths of a map network.

Definition 2. *Consider a path $p = (i_0, \cdots, i_k)$ along $\mathcal{G}$. We define the composite map along p induced from a map network $\mathcal{F}$ on $\mathcal{G}$ as*

$$f_p = f_{i_{k-1} i_k} \circ \cdots \circ f_{i_0 i_1}. \tag{1}$$

In particular, we define $f_\varnothing := I$ where $\varnothing$ can refer to any self-loop.

In the remaining text, for two successive paths p and q, we use $p \sim q$ to denote their composition.

Now we state the path-invariance constraint for map networks.

Definition 3. *Let $\mathcal{G}_{\text{path}}(u, v)$ collect all paths in $\mathcal{G}$ that connect u to v. We define the set of all possible path pairs of $\mathcal{G}$ as*

$$\mathcal{G}_{\text{pair}} = \bigcup_{u,v \in \mathcal{V}} \{(p, q) | p, q \in \mathcal{G}_{\text{path}}(u, v)\}.$$

We say $\mathcal{F}$ is path-invariant if

$$f_p = f_q, \qquad \forall (p, q) \in \mathcal{G}_{\text{pair}}. \tag{2}$$

Remark 1. Since $\mathcal{G}_{\text{path}}(v, v)$ collects the self-loop at each v, it is easy to check that path-invariance induces cycle-consistency (c.f.[15]). On the other hand, for undirected map networks it is easy to see that cycle-consistency induces path-invariance. However, this property is not true for directed map networks. For example, a map network with three vertices $\{a, b, c\}$ and three directed maps f_{ab}, f_{bc}, and f_{ac} has no-cycle, but one path pair $(f_{bc} \circ f_{ab}, f_{ac})$.

3.2. Path-Invariance Basis

A challenge of enforcing the path-invariant constraint is that there are many possible paths between each pair of domains in a graph, leading to an intractable number of path pairs. This raises the question of how to compute a path-invariance basis $\mathcal{B} \subset \mathcal{G}_{\text{pair}}$, which is a set of independent path pairs that are sufficient for enforcing the path-invariance property of any map network $\mathcal{F}$. To rigorously define path-invariance basis, we introduce three primitive operations on path pairs merge, stitch and cut(See Figure 2):

Definition 4. *Consider a directed graph $\mathcal{G}$. We say two path pairs (p, q) and (p', q') are compatible if one path in $\{p, q\}$ is a sub-path of one path in $\{p', q'\}$ or vice-versa. Without losing generality, suppose p is a sub-path of p' and we write $p' = r \sim p \sim r'$, which stitches three sub-paths r,p,and r' in order. We define the* merge *operation so that it takes two compatible path pairs (p, q) and $(r \sim p \sim r', q')$ as input and outputs a new path pair $(r \sim q \sim r', q')$.*

We proceed to define the stitch operation:

Definition 5. *We define the* stitch *operation so that it takes as input two path pairs $(p, q), p, q \in \mathcal{G}_{\text{path}}(u, v)$ and $(p', q'), p', q' \in \mathcal{G}_{\text{path}}(v, w)$ and outputs $(p \sim p', q \sim q')$.*

Finally we define the cut operation on two cycles, which will be useful for strongly connected graphs:

Definition 6. *Operation* cut *takes as input two path pairs $(\mathcal{C}_1, \varnothing)$ and $(\mathcal{C}_2, \varnothing)$ where $\mathcal{C}_1$ and $\mathcal{C}_2$ are two distinct cycles that have two common vertices u, v and share a common path from v to u. Specifically, we assume these two cycles*

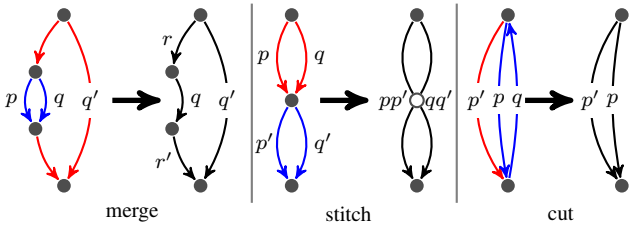


Figure 2: Illustrations of Operations

are $u \xrightarrow{p} v \xrightarrow{q} u$ *and* $u \xrightarrow{p'} v \xrightarrow{q} u$ *where* $p, p' \in \mathcal{G}_{\text{path}}(u, v)$ *and* $q \in \mathcal{G}_{\text{path}}(v, u)$. *We define the output of the* cut *operation as a new path pair* (p, p').

The Definition 6 is necessary because $f_p \circ f_q = f_{p'} \circ f_q = I$ implies $f_p = f_{p'}$. As we will see later, this operation is useful for deriving new path-invariance basis.

Now we define path-invariance , which is the critical concept of this paper:

Definition 7. *We say a collection of path pairs* $\mathcal{B} = \{(p, q)\}$ *is a path-invariance basis on* $\mathcal{G}$ *if every path-pair* $(p, q) \in \mathcal{G}_{\text{pair}} \setminus \mathcal{B}$ *can be induced from a subset of* $\mathcal{B}$ *through a series of* merge, stitch *and/or* cut *operations.*

The following proposition shows the importance of path-invariance basis:

Proposition 1. *Consider a path-invariance basis* $\mathcal{B}$ *of a graph* $\mathcal{G}$. *Then for any map network* $\mathcal{F}$ *on* $\mathcal{G}$, *if*

$$f_p = f_q, \quad (p, q) \in \mathcal{B},$$

then $\mathcal{F}$ *is path-invariant.*

3.3. Path-Invariance Basis Computation

We first discuss the criteria for path-invariance basis computation. Since we will formulate a loss term for each path pair in a path-invariance basis to enforce the path-invariance constraint of a map network, we place two objectives for computing a path-invariance basis. First, we require the length of the paths in each path pair to be small. Intuitively, enforcing consistency between long paths weakens the regularization on each involved map. Second, we want the size of the resulting path-invariance basis to be small in order to increase the effectiveness of gradient-descent based optimization strategies. Note that unlike cycle bases that have a fixed size (c.f. [22]), the sizes of path-invariance bases vary. In fact, in the worst case, the size of a path-invariance basis may be exponential in $|\mathcal{V}|$.

In the following, we will present an algorithm that is guaranteed to return a path-invariance basis whose size is polynomial in $|\mathcal{V}|$, i.e., $O(|\mathcal{V}||\mathcal{E}|)$ in the worst case. Our algorithm builds upon the classical result that a directed graph $\mathcal{G}$ can be factored into a directed acyclic graph whose vertices are strongly connected components of $\mathcal{G}$ (c.f. [4]). In light of this, we describe our algorithm in three steps. We first show how to compute a path-invariance basis for a directed acyclic graph. We then discuss the case of strongly connected components. Finally, we show how to extend the result of the first two settings to arbitrary directed graphs.

Directed acyclic graph (or DAG). Our algorithm utilizes an important property that every DAG admits a topological order of vertices that are consistent with the edge orientations (c.f. [4]). Specifically, consider a DAG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. A topological order is a bijection $\sigma : \{1, \cdots, |\mathcal{V}|\} \to \mathcal{V}$ so that we have $\sigma^{-1}(u) < \sigma^{-1}(v)$ whenever $(u, v) \in \mathcal{E}$. A topological order of a DAG can be calculated by Tarjan's algorithm (c.f. [46]).

Algorithm 1 The high level algorithm flow to find a path-invariance basis.

input: Directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.
output: Path-invariance basis $\mathcal{B}$.

1: Calculate SCCs $\mathcal{G}_1, \ldots, \mathcal{G}_K$ for $\mathcal{G}$ and the resulting contracted DAG $\mathcal{G}_{dag}$.
2: Calculate a path-invriance basis $\mathcal{B}_{dag}$ for $\mathcal{G}_{dag}$ and transform $\mathcal{B}_{dag}$ to $\overline{\mathcal{B}}_{dag}$ whose elements are path pairs on $\mathcal{G}$.
3: Calculate a path-invariance basis $\mathcal{B}_i$ for $\mathcal{G}_i$.
4: Calculate path-invirance pairs $\mathcal{B}_{ij}$ whenever $\mathcal{G}_i$ can reach $\mathcal{G}_j$ in $\mathcal{G}_{dag}$.
5: **return** $\mathcal{B} = \overline{\mathcal{B}}_{dag} \bigcup \left(\cup_{i=1}^{K} \mathcal{B}_i \right) \bigcup \left(\cup_{ij} \mathcal{B}_{ij} \right)$

Our algorithm starts with a current graph $\mathcal{G}_{cur} = (\mathcal{V}, \varnothing)$ to which we add all edges in $\mathcal{E}$ in some order later. Specifically, the edges in $\mathcal{E}$ will be visited with respect to a (partial) edge order $\prec$ where $\forall (u, v), (u', v') \in \mathcal{E}$, $(u, v) \prec (u', v')$ if and only if $\sigma^{-1}(v) < \sigma^{-1}(v')$. Note that two edges (u, v), (u', v) with the same head can be in arbitrary order.

For each newly visited edge $(u, v) \in \mathcal{E}$, we collect a set of candidate vertices $\mathcal{P} \subset \mathcal{V}$ such that every vertex $w \in \mathcal{P}$ can reach both u and v in $\mathcal{G}_{cur}$. Next we construct a set $\overline{\mathcal{P}}$ by removing from $\mathcal{P}$ all $w \in \mathcal{P}$ such that w can reach some distinct $w' \in \mathcal{P}$. In other words, w is redundant because of w' in this case. For each vertex $w \in \overline{\mathcal{P}}$, we collect a new path-pair $(p', p \sim uv)$, where p and p' are shortest paths from w to u and v, respectively. After collecting path pairs, we augment $\mathcal{G}_{cur}$ with (u, v). With $\mathcal{B}_{dag}(\sigma)$ we denote the resulting path-pair set after $\mathcal{E}_{cur} = \mathcal{E}$.

Theorem 3.1. *Every topological order* σ *of* $\mathcal{G}$ *returns a path-invariance basis* $\mathcal{B}_{dag}(\sigma)$ *whose size is at most* $|\mathcal{V}||\mathcal{E}|$.

Strongly connected graph (or SCG). To construct a path-invariance basis of a SCG $\mathcal{G}$, we run a slightly-modified depth-first search on $\mathcal{G}$ from arbitrary vertex. Since $\mathcal{G}$ is strongly connected, the resulting spanning forest must be a tree, denoted by $\mathcal{T}$. The path pair set $\mathcal{B}$ is the result we obtain. In addition, we use a $\mathcal{G}_{dag}$ to collect a acyclic subgraph of $\mathcal{G}$ and initially it is set as empty. When traversing edge (u, v), if v is visited for the first time, then we add (u, v) to both $\mathcal{T}$ and $\mathcal{G}_{dag}$. Otherwise, there can be two possible cases:

- v is an ancestor of u in $\mathcal{T}$. In this case we add cycle consistency $(P \sim (u, v), \varnothing)$, where P is the tree path from v to u, into $\mathcal{B}$.
- Otherwise, add (u, v) into $\mathcal{G}_{dag}$.

It can be proved that $\mathcal{G}_{dag}$ is indeed an acyclic graph (See Appendix A.3). Thus we can obtain a path-invariance basis

on $\mathcal{G}_{dag}$ by running the algorithm stated in the DAG case. We add this basis into $\mathcal{B}$. The following proposition ensures $\mathcal{B}$ is a path-invraiance basis of $\mathcal{G}$

Proposition 2. *The path pair set $\mathcal{B}$ constructed above is a path-invariance basis of $\mathcal{G}$.*

General directed graph. Given path-invariance bases constructed on DAGs and SCGs, constructing path-invariance bases on general graphs is straight-forward. Specifically, consider strongly connected components $\mathcal{G}_i, 1 \leq i \leq K$ of a graph $\mathcal{G}$. With $\mathcal{G}_{dag}$ we denote the directed acylic graph among $\mathcal{G}_i, 1 \leq i \leq K$. We first construct path-invariance bases $\mathcal{B}_{dag}$ and $\mathcal{B}_i$ for $\mathcal{G}_{dag}$ and each $\mathcal{G}_i$, respectively. We then construct a path-invariance basis $\mathcal{B}$ of $\mathcal{G}$ by collecting three groups of path pairs. The first group simply combines $\mathcal{B}_i, 1 \leq i \leq K$. The second group extends $\mathcal{B}_{dag}$ to the original graph. This is done by replacing each edge $(\mathcal{G}_i, \mathcal{G}_j) \in \mathcal{E}_{dag}$ through a shortest path on $\mathcal{G}$ that connects the representatives of $\mathcal{G}_i$ and $\mathcal{G}_j$ where representatives are arbitrarily chosen at first for each component. To calculate the third group, consider all oriented edges between each $(\mathcal{G}_i, \mathcal{G}_j) \in \mathcal{E}_{dag}$:

$$\mathcal{E}_{ij} = \{uv \in \mathcal{E} : u \in \mathcal{V}_i, v \in \mathcal{V}_j\}.$$

Note that when constructing $\mathcal{B}_{dag}$, all edges in $\mathcal{E}_{ij}$ are shrinked to one edge in $\mathcal{E}_{dag}$. This means when constructing $\mathcal{B}$, we have to enforce the consistency among $\mathcal{E}_{ij}$ on the original graph $\mathcal{G}$. This can be done by constructing a tree $\mathcal{T}_{ij}$ where $\mathcal{V}(\mathcal{T}_{ij}) = \mathcal{E}_{ij}$, $\mathcal{E}(\mathcal{T}_{ij}) \subset \mathcal{E}_{ij}^2$. $\mathcal{T}_{ij}$ is a minimum spanning tree on the graph whose vertex set is $\mathcal{E}_{ij}$ and the weight associated with edge $(uv, u'v') \in \mathcal{E}_{ij}^2$ is given by the sum of lengths of $\widehat{uu'}$ and $\widehat{vv'}$. This strategy encourages reducing the total length of the resulting path pairs in $\mathcal{B}_{ij}$ that will be defined below:

$$\mathcal{B}_{ij} := \{(\widehat{uu'} \sim u'v', uv \sim \widehat{vv'}) : (uv, u'v') \in \mathcal{E}(\mathcal{T}_{ij})\},$$

where $\widehat{uu'}$ and $\widehat{vv'}$ denote the shortest paths from u to u' on $\mathcal{G}_i$ and from v to v' on $\mathcal{G}_j$, respectively.Algorithm 1 shows the path-invariance basis of a DAG computed using the algorithm described above.

Theorem 3.2. *The path-pairs $\mathcal{B}$ derived from $\mathcal{B}_{dag}$, $\{\mathcal{B}_i : 1 \leq i \leq K\}$, and $\{\mathcal{B}_{ij} : (\mathcal{G}_i, \mathcal{G}_j) \in \mathcal{E}_{dag}\}$ using the algorithm described above is a path-invariance basis for $\mathcal{G}$.*

Proposition 3. *The size of $\mathcal{B}$ is upper bounded by $|\mathcal{V}||\mathcal{E}|$*[1].

4. Joint Map Network Optimization

In this section, we present a formulation for jointly optimizing a map network using the path-variance basis computed in the preceding section.

[1]We conjecture that computing the path-invariance basis with minimum size is NP-hard.

Consider the map network defined in Def. 1. We assume the map associated with each edge $(i, j) \in \mathcal{E}$ is a parametric map $f_{ij}^{\theta_{ij}}$, where θ_{ij} denotes hyper-parameters or network parameters of f_{ij}. We assume the supervision of map network is given by a superset $\overline{\mathcal{E}} \supset \mathcal{E}$. As we will see later, such instances happen when there exist paired data between two domains, but we do not have a direct neural network between them. To utilize such supervision, we define the induced map along an edge $(i, j) \in \overline{\mathcal{E}}$ as the composition map (defined in (1)) $f_{\widehat{v_i v_j}}^{\Theta}$ along the short path $\widehat{v_i v_j}$ from v_i to v_j. Here $\Theta = \{\theta_{ij}, (i, j) \in \mathcal{E}\}$ collects all the parameters. We define each supervised loss term as $l_{ij}(f_{ij}^{\Theta}), \forall (i, j) \in \overline{\mathcal{E}}$. The specific definition of l_{ij} will be deferred to Section 5.

Besides the supervised loss terms, the key component of joint map network optimization utilizes a self-supervision loss induced from the path-invariance basis $\mathcal{B}$. Let $d_{\mathcal{D}_i}(\cdot, \cdot)$ be a distance measure associated with domain $\mathcal{D}_i$. Consider an empirical distribution P_i of $\mathcal{D}_i$. We define the total loss objective for joint map network optimization as

$$\min_{\Theta} \sum_{(i,j)\in\overline{\mathcal{E}}} l_{ij}(f_{\widehat{v_i v_j}}^{\Theta}) + \lambda \sum_{(p,q)\in\mathcal{B}} \mathop{E}_{v\sim P_{p_t}} d_{\mathcal{D}_{p_t}}(f_p^{\Theta}(v), f_q^{\Theta}(v)) \tag{3}$$

where p_t denotes the index of the end vertex of p. Essentially, (3) combines the supervised loss terms and an unsupervised regularization term that ensures the learned representations are consistent when passing unlabeled instances across the map network. We employ the ADAM optimizer [27] for optimization. In addition, we start with a small value of λ, e.g., $\lambda = 10^{-2}$, to solve (3) for 40 epochs. We then double the value of λ every 10 epochs. We stop the training procedure when $\lambda \geq 10^3$. The training details are deferred to the Appendix.

5. Experimental Evaluation

This section presents an experimental evaluation of our joint map network optimization framework across three settings, namely, shape matching (Section 5.1), dense image maps (Section 5.2), and a network of hybrid 3D representations for 3D semantic segmentation (Section 5.3).

5.1. Map Network of Shape Maps

We begin with the task of joint shape matching [31, 25, 15, 16, 10], which seeks to jointly optimize a network of shape maps to improve the initial maps computed between pairs of shapes in isolation. We utilize the functional map representation described in [33, 47, 16]. Specifically, each domain $\mathcal{D}_i$ is given by a linear space spanned by the leading m eigenvectors of a graph Laplacian [16] (we choose $m = 30$ in our experiments). The map from $\mathcal{D}_i$ to $\mathcal{D}_j$ is given by a matrix $X_{ij} \in \mathbb{R}^{m\times m}$. Let $\mathcal{B}$ be a path-invariance basis for the associated graph $\mathcal{G}$. Adapting (3), we solve the

following optimization problem for joint shape matching:

$$\sum_{(i,j)\in\mathcal{E}} \|X_{ij} - X_{ij}^{in}\|_1 + \lambda \sum_{(p,q)\in\mathcal{B}} \|X_p - X_q\|_{\mathcal{F}}^2 \quad (4)$$

where $\|\cdot\|_1$ and $\|\cdot\|_{\mathcal{F}}$ are the element-wise L1-norm and the matrix Frobenius norm, respectively. X_{ij}^{in} denotes the initial functional map converted from the corresponding initial shape map associated with edge (i,j) using [33].

Dataset. We perform experimental evaluation on SHREC07–Watertight [13], which is a challenging dataset for evaluating shape maps. Specifically, SHREC07-Watertight contains 400 shapes across 20 categories. Among them, we choose 11 categories (i.e., Human, Glasses, Airplane, Ant, Teddy, Hand, Plier, Fish, Bird, Armadillo, Fourleg) that are suitable for inter-shape mapping. We also test our approach on two large-scale datasets Aliens (200 shapes) and Vase (300 shapes) from ShapeCOSEG [50]. For initial maps, we employ blended intrinsic maps [26], a state-of-the-art method for shape matching. We test our approach under two graphs $\mathcal{G}$. The first graph is a clique graph. The second graph connects each shape with k-nearest neighbor with respect to the GMDS descriptor [42] ($k = 10$ in our experiments).

Baseline approaches and evaluation metric. We compare our approach to five baseline approaches, including three state-of-the-art approaches and two variants of our approach. Three state-of-the-art approaches are 1) functional-map based low-rank matrix recovery [16], 2) point-map based low-rank matrix recovery via alternating minimization [58], and 3) consistent partial matching via sparse modeling [10]. Two variants are 4) using a set of randomly sampled cycles [54] whose size is the same as $|\mathcal{B}|$, and 5) using the path-invariance basis derived from the fundamental cycle-basis of $\mathcal{G}$ (c.f. [22]) (which may contain long cycles).

We evaluate the quality of each map through annotated key points (Please refer to the appendix). Following [26, 15, 16], we report the cumulative distribution function (or CDF) of geodesic errors of predicted feature correspondences.

Analysis of results. Figure 3 shows CDFs of our approach and baseline approaches. All participating methods exhibit considerable improvements from the initial maps, demonstrating the benefits of joint shape matching. Compared to state-of-the-art approaches, our approach is comparable when $\mathcal{G}$ is a clique and exhibits certain performance gains when $\mathcal{G}$ is sparse. One explanation is that low-rank approaches are based on relaxations of the cycle-consistency constraint (c.f. [15]), and such relaxations become loose on sparse graphs. In contrast, our approach explicitly enforces the cycle-consistency constraint (through the generalized path-invariance constraint). Compared to two variants of our approach, our approach delivers the best results on both clique graphs and knn-graphs. This is because the two alternative strategies generate many long paths and cycles in $\mathcal{B}$, making the total objective function (3) hard to optimize.

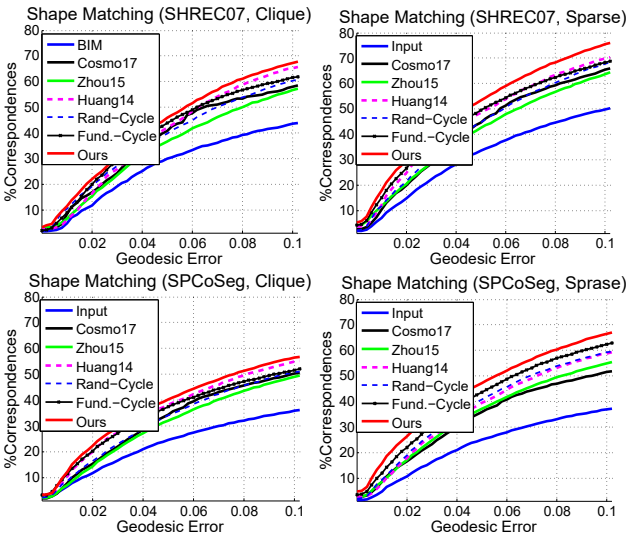


Figure 3: Baseline comparison on benchmark datasets. We show cumulative distribution functions (or CDFs) of each method with respect to annotated feature correspondences.

On knn-graphs, both our approach and the baseline of using the fundamental cycle-basis outperform the baseline of randomly sampling path pairs, showing the importance of computing a path-invariance basis for enforcing the consistency constraint.

5.2. Map Network of Dense Image Maps

In the second setting, we consider the task of optimizing dense image flows across a collection of relevant images. We again model this task using a map network $\mathcal{F}$, where each domain $\mathcal{D}_i$ is given by an image I_i. Our goal is to compute a dense image map $f_{ij} : I_i \to I_j$ (its difference to the identity map gives a dense image flow) between each pair of input images. To this end, we precompute initial dense maps $f_{ij}^{in}, \forall (i,j) \in \mathcal{E}$ using DSP [24], which is a state-of-the-art approach for dense image flows. Our goal is to obtain improved dense image maps $f_{ij}, \forall (i,j) \in \mathcal{E}$, which lead to dense image maps between all pairs of images in $\mathcal{F}$ via map composition (See (1)). Due to scalability issues, state-of-the-art approaches for joint estimation of dense image flows [28, 23, 36, 57] are limited to a small number of relatively low-resolution images. To address this issue, we encode dense image maps using the neural network f^θ described in [56]. Given a fixed map network $\mathcal{F}$ and the initial dense maps $f_{ij}^{in}, (i,j) \in \mathcal{E}$, we formulate a similar optimization problem as (4) to learn the network parameters θ:

$$\min_\theta \sum_{(i,j)\in\mathcal{E}} \|f_{ij}^\theta - f_{ij}^{in}\|_1 + \lambda \sum_{(p,q)\in\mathcal{B}} \|f_p^\theta - f_q^\theta\|_{\mathcal{F}}^2 \quad (5)$$

where $\mathcal{B}$ denotes a path-invariance basis associated with $\mathcal{F}$; p_s is the index of the start vertex of p; f_p^θ is the composite network along path p.

	aero	bike	boat	bottle	bus	car	chair	table	mbike	sofa	train	tv	mean
Congealing	0.13	0.24	0.05	0.21	0.22	0.11	0.09	0.05	0.14	0.09	0.10	0.09	0.13
RASL	0.18	0.20	0.05	0.36	0.33	0.19	0.14	0.06	0.19	0.13	0.14	0.29	0.19
CollectionFlow	0.17	0.18	0.06	0.33	0.31	0.15	0.15	0.04	0.12	0.11	0.10	0.11	0.12
DSP	0.19	0.33	0.07	0.21	0.36	0.37	0.12	**0.07**	0.19	0.13	0.15	0.21	0.20
FlowWeb	0.31	0.42	**0.08**	0.37	0.56	0.51	0.12	0.06	0.23	0.18	0.19	0.34	0.28
Ours-Dense	0.29	0.42	0.07	0.39	0.53	0.55	0.11	0.06	0.22	0.18	0.21	0.31	0.28
Ours-Undirected	0.32	0.43	0.07	0.43	0.56	0.55	0.18	0.06	0.26	0.21	**0.25**	0.37	0.31
Ours	**0.35**	**0.45**	0.07	**0.45**	**0.63**	**0.62**	**0.19**	0.06	**0.27**	**0.22**	0.23	**0.38**	**0.33**

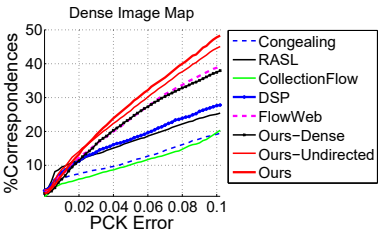


Figure 4: (Left) Keypoint matching accuracy (PCK) on 12 rigid PASCAL VOC categories ($\alpha = 0.05$). Higher is better. (Right) Plots of the mean PCK of each method with varying α

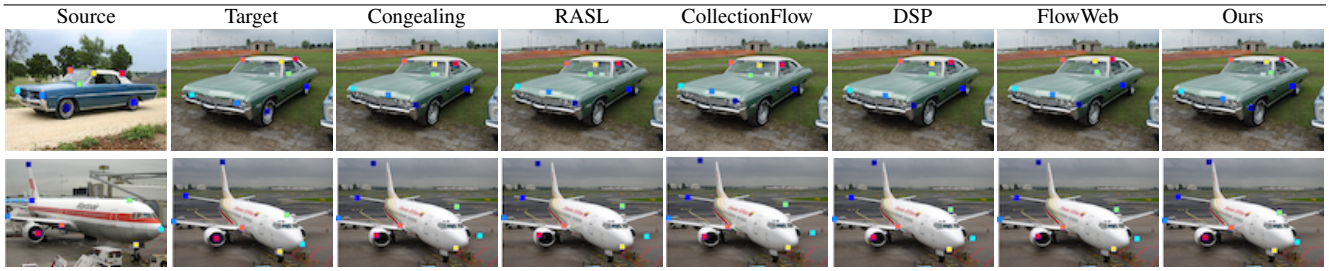


Figure 5: Visual comparison between our approach and state-of-the-art approaches. This figure is best viewed in color, zoomed in. More examples are included in the appendix.

Dataset. The image sets we use are sampled from 12 rigid categories of the PASCAL-Part dataset [6]. To generate image sets that are meaningful to align, we pick the most popular view for each category (who has the smallest variance among 20-nearest neighbors). We then generate an image set for that category by collecting all images whose poses are within 30° of this view. We construct the map network by connecting each image with k-nearest neighbors with respect to the DSP matching score [24]. Note that the resulting $\mathcal{F}$ is a directed graph as DSP is directed.

Baseline approaches and evaluation metric. We compare our approach with Congealing [28], Collection Flow [23], RASL [36], and FlowWeb [57]. We use publicly available code for all baselines except Collection Flow, for which we implement our own version in Matlab. Note that both Flowweb and our approach use DSP as input. Moreover, we did not compare to [56], since it uses additional synthetic images as supervision. To run baseline approaches, we follow the protocol of [57] to further break each dataset into smaller ones with maximum size of 100. In addition, we consider two variants of our approach: Ours-Dense and Ours-Undirected. Ours-Dense uses the clique graph for $\mathcal{F}$. Ours-Undirected uses an undirected knn-graph, where the weight of each edge averages the bi-directional DSP matching scores (c.f. [24]). We employ the standard PCK measure [51], which reports the percentage of keypoints whose prediction errors fall within $\alpha \cdot \max(h, w)$ (h and w are image height and width respectively).

Analysis of results. As shown in Figure 4 and Figure 5, our approach outperforms all existing approaches across most of the categories. Several factors contribute to such improvements. First, our approach can jointly optimize more images than baseline approaches and thus benefits more from the data-driven effect of joint matching [15, 7]. This explains why all variants of our approach are either comparable or superior to baseline approaches. Second, our approach avoids fitting a neural network directly to dissimilar images and focuses on relatively similar images (other maps are generated by map composition), leading to additional performance gains. In fact, all existing approaches, which operate on sub-groups of similar images, also implicitly benefit from map composition. This explains why FlowWeb exhibits competing performance against Ours-Dense. Finally, Ours-Directed is superior to Ours-Undirected. This is because the outlier-ratio of f_{ij}^{in} in Ours-Undirected is higher than that of Ours-Directed, which selects edges purely based on matching scores.

5.3. Map Network of 3D Representations

In the third setting, we seek to jointly optimize a network of neural networks to improve the performance of individual networks. We are particularly interested in the task of semantic segmentation of 3D scenes. Specifically, we consider a network with seven 3D representations (See Figure 1). The first representation is the input mesh. The last representation is the space of 3D semantic segmentations. The second to fourth 3D representations are point clouds with different number of points: PCI (12K), PCII (8K), and PCIII(4K). The motivation of varying the number of points is that the patterns learned under different number of points show certain variations, which are beneficial to each other. In a similar fashion, the fifth and sixth are volumetric representations under two resolutions: VOLI($32 \times 32 \times 32$) and VOLII($24 \times 24 \times 24$). The directed maps between differ-

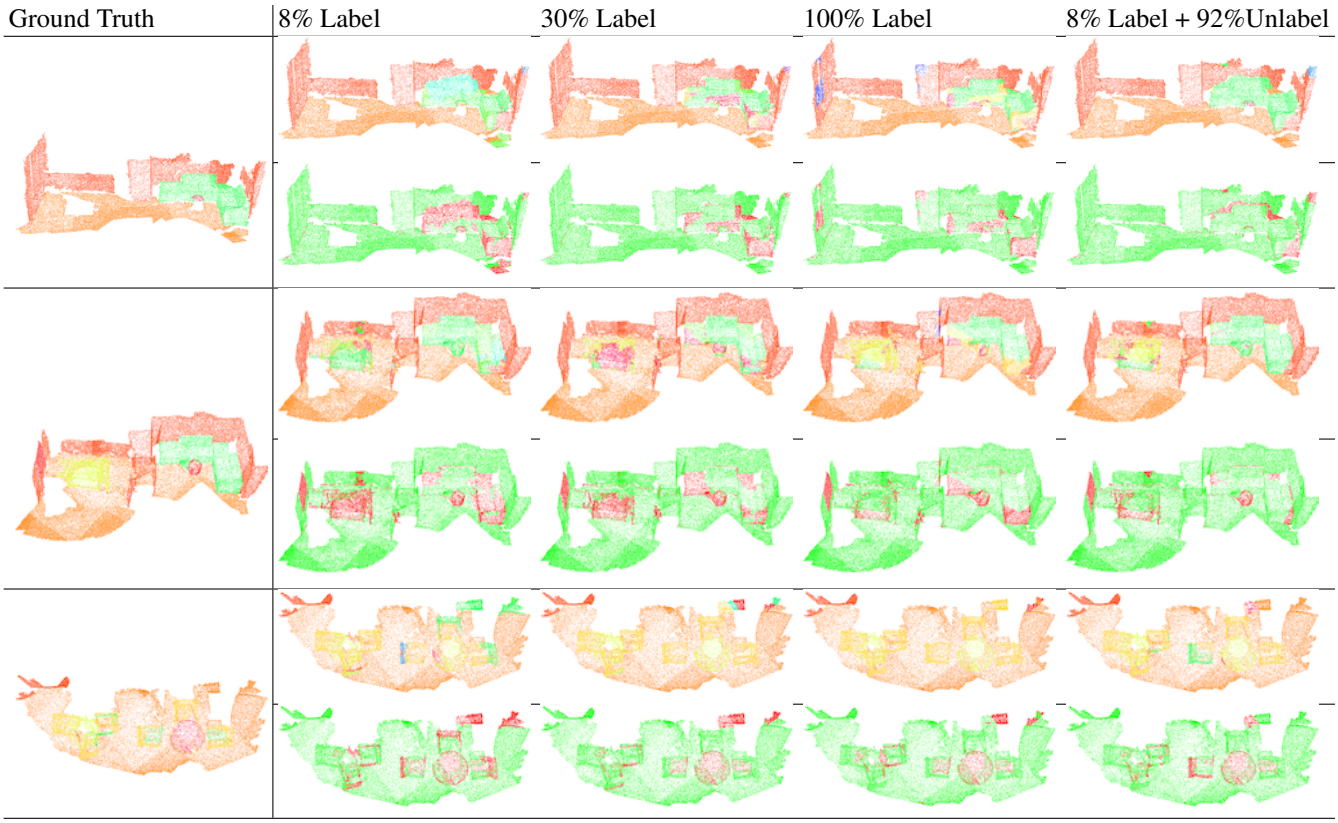


Figure 6: Qualitative comparisons of 3D semantic segmentation results on ScanNet [12]. Each row represents one testing instance, where ground truth and top sub-row show prediction for 21 classes and bottom sub-row only shows correctly labeled points. (Green indicates correct predictions, while red indicates false predictions.) This figure is best viewed in color, zoomed in.

ent 3D representations fall into three categories, which are summarized below:

1. Semantic segmentation networks. Each point cloud or volumetric representation is associated with a segmentation network. Specifically, we use PointNet++ [37] and 3D U-Net[9], which are state-of-the-art network architectures for point cloud and volumetric representations, respectively.

2. Pointcloud sub-sampling maps. We have six pointcloud sub-sampling maps among the mesh representation (we uniformly sample 24K points using [32]) and three point cloud representations. For each point sub-sampling map, we force the down-sampled point cloud to align with the feature points of the input point cloud [35]. Note that this down-sampled point cloud is also optimized through a segmentation network to maximize the segmentation accuracy.

3. Generating volumetric representations. Each volumetric representation is given by the signed-distance field (or SDF) described in [45]. These SDFs are precomputed.

Experimental setup. We have evaluated our approach on ScanNet semantic segmentation benchmark [12]. Our goal is to evaluate the effectiveness of our approach when using a small labeled dataset and a large unlabeled dataset. To this end, we consider three baseline approaches, which train the segmentation network under each individual representation using 100%, 30%, and 8% of the labeled data. We then test our approach by utilizing 8% of the labeled data, which defines the data term in (3), and 92% of the unlabeled data (only the root has an empirical distribution), which defines the regularization term of (3). We initialize the segmentation network for point clouds using uniformly sampled points trained on labeled data. We then fine-tune the entire network using both labeled and unlabeled data. Code is publicly available at https://github.com/zaiweizhang/path_invariance_map_network.

	PCI	PCII	PCIII	VOLI	VOLII	Ensm
100% Label (Isolated)	84.2	83.3	83.4	81.9	81.5	85
8% Label (Isolated)	79.2	78.3	78.4	78.7	77.4	81.4
8% Label + Unlabel (Joint)	82.3	82.5	82.3	81.6	79.0	83.4
30% Label (Isolated)	80.8	81.9	81.2	80.3	79.5	83.2

Table 1: Semantic surface voxel label prediction accuracy on ScanNet test scenes (in percentages), following [37]. We also show the ensembled prediction accuracy with five representations in the last column.

Analysis of results. Figure 6 and Table 1 present qualitative and quantitative comparisons between our approach and baselines. Across all 3D representations, our approach leads to consistent improvements, demonstrating the robust-

ness of our approach. Specifically, when using 8% labeled data and 92% unlabeled data, our approach achieved competing performance as using 30% to 100% labeled data when trained on each individual representation. Moreover, the accuracy on VOLI is competitive against using 100% of labeled data, indicating that the patterns learned under the point cloud representations are propagated to train the volumetric representations. We also tested the performance of applying popular vote [39] on the predictions of using different 3D representations. The relative performance gains of using different configurations of training data remain similar (See the last column in Table1). Please refer to Appendix C for more experimental evaluations and baseline comparisons.

6. Conclusions

In this paper, we have studied the problem of optimizing a directed map network. We have introduced the path-invariance constraint, which can be effectively encoded using path-invariance bases. We have described an algorithm for computing a path-invariance basis with polynomial time and space complexities. The effectiveness of this approach is demonstrated on three groups of map networks with diverse applications.

References

[1] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski. Building rome in a day. *Commun. ACM*, 54(10):105–112, Oct. 2011. 1

[2] F. Arrigoni, A. Fusiello, B. Rossi, and P. Fragneto. Robust rotation synchronization via low-rank and sparse matrix decomposition. *CoRR*, abs/1505.06079, 2015. 2

[3] C. Bajaj, T. Gao, Z. He, Q. Huang, and Z. Liang. Smac: Simultaneous mapping and clustering via spectral decompositions. In *ICML*, pages 100–108, 2018. 1

[4] J. Bang-Jensen and G. Z. Gutin. *Digraphs - theory, algorithms and applications*. Springer, 2002. 4

[5] A. Chatterjee and V. M. Govindu. Efficient and robust large-scale rotation averaging. In *ICCV*, pages 521–528. IEEE Computer Society, 2013. 1, 2

[6] X. Chen, R. Mottaghi, X. Liu, S. Fidler, R. Urtasun, and A. L. Yuille. Detect what you can: Detecting and representing objects using holistic models and body parts. *CoRR*, abs/1406.2031, 2014. 7

[7] Y. Chen, L. J. Guibas, and Q. Huang. Near-optimal joint object matching via convex relaxation. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 100–108, 2014. 1, 2, 7

[8] M. Cho, S. Kwak, C. Schmid, and J. Ponce. Unsupervised object discovery and localization in the wild: Part-based matching with bottom-up region proposals. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 1201–1210, 2015. 1

[9] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 424–432. Springer, 2016. 8, 15

[10] L. Cosmo, E. Rodolà, A. Albarelli, F. Mémoli, and D. Cremers. Consistent partial matching of shape collections via sparse modeling. *Comput. Graph. Forum*, 36(1):209–221, 2017. 5, 6

[11] D. J. Crandall, A. Owens, N. Snavely, and D. P. Huttenlocher. Sfm with mrfs: Discrete-continuous optimization for large-scale structure from motion. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(12):2841–2853, 2013. 1

[12] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Niener. Scannet: Richly-annotated 3d reconstructions of indoor scenes, 2017. cite arxiv:1702.04405. 1, 2, 8, 15, 18

[13] D. Giorgi, S. Biasotti, and L. Paraboschi. Shape retrieval contest 2007: Watertight models track, 2007. 6

[14] Q. Huang, Y. Chen, and L. J. Guibas. Scalable semidefinite relaxation for maximum A posterior estimation. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 64–72, 2014. 2

[15] Q. Huang and L. Guibas. Consistent shape maps via semidefinite programming. In *Proceedings of the Eleventh Eurographics/ACMSIGGRAPH Symposium on Geometry Processing*, pages 177–186, 2013. 1, 2, 3, 5, 6, 7

[16] Q. Huang, F. Wang, and L. J. Guibas. Functional map networks for analyzing and exploring large shape collections. *ACM Trans. Graph.*, 33(4):36:1–36:11, 2014. 1, 2, 5, 6

[17] Q. Huang, G. Zhang, L. Gao, S. Hu, A. Butscher, and L. J. Guibas. An optimization approach for extracting and encoding consistent maps in a shape collection. *ACM Trans. Graph.*, 31(6):167:1–167:11, 2012. 1

[18] Q.-X. Huang, S. Flöry, N. Gelfand, M. Hofer, and H. Pottmann. Reassembling fractured objects by geometric matching. *ACM Trans. Graph.*, 25(3):569–578, July 2006. 1, 2

[19] X. Huang, Z. Liang, C. Bajaj, and Q. Huang. Translation synchronization via truncated least squares. In *NIPS*, page to appear, 2017. 1, 2

[20] D. F. Huber and M. Hebert. Fully automatic registration of multiple 3d data sets. *Image and Vision Computing*, 21:637–650, 2001. 1, 2

[21] M. Johnson, M. Schuster, Q. V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. B. Viégas, M. Wattenberg, G. Corrado, M. Hughes, and J. Dean. Google's multilingual neural machine translation system: Enabling zero-shot translation. *CoRR*, abs/1611.04558, 2016. 2

[22] T. Kavitha, C. Liebchen, K. Mehlhorn, D. Michail, R. Rizzi, T. Ueckerdt, and K. A. Zweig. Survey: Cycle bases in graphs characterization, algorithms, complexity, and applications. *Comput. Sci. Rev.*, 3(4):199–243, Nov. 2009. 3, 4, 6

[23] I. Kemelmacher-Shlizerman and S. M. Seitz. Collection flow. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1792–1799. IEEE, 2012. 6, 7

[24] J. Kim, C. Liu, F. Sha, and K. Grauman. Deformable spatial pyramid matching for fast dense correspondences. In *CVPR*, pages 2307–2314. IEEE Computer Society, 2013. 1, 6, 7

[25] V. G. Kim, W. Li, N. J. Mitra, S. DiVerdi, and T. Funkhouser. Exploring collections of 3d models using fuzzy correspondences. *ACM Trans. Graph.*, 31(4):54:1–54:11, July 2012. 1, 5

[26] V. G. Kim, Y. Lipman, and T. Funkhouser. Blended intrinsic maps. In *ACM SIGGRAPH 2011 Papers*, SIGGRAPH '11, pages 79:1–79:12, New York, NY, USA, 2011. ACM. 6, 15

[27] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 5, 15

[28] E. G. Learned-Miller. Data driven image models through continuous joint alignment. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(2):236–250, Feb. 2006. 6, 7

[29] S. Leonardos, X. Zhou, and K. Daniilidis. Distributed consistent data association via permutation synchronization. In *ICRA*, pages 2645–2652. IEEE, 2017. 1

[30] C. Liu, J. Yuen, and A. Torralba. Sift flow: Dense correspondence across scenes and its applications. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5):978–994, May 2011. 1

[31] A. Nguyen, M. Ben-Chen, K. Welnicka, Y. Ye, and L. J. Guibas. An optimization approach to improving collections of shape maps. *Comput. Graph. Forum*, 30(5):1481–1491, 2011. 1, 2, 5

[32] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin. Shape distributions. *ACM Trans. Graph.*, 21(4):807–832, Oct. 2002. 8

[33] M. Ovsjanikov, M. Ben-Chen, J. Solomon, A. Butscher, and L. Guibas. Functional maps: A flexible representation of maps between shapes. *ACM Transactions on Graphics*, 31(4), 2012. 5, 6

[34] D. Pachauri, R. Kondor, and V. Singh. Solving the multi-way matching problem by permutation synchronization. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 1860–1868. Curran Associates, Inc., 2013. 2

[35] M. Pauly, R. Keiser, and M. Gross. Multi-scale Feature Extraction on Point-Sampled Surfaces. *Computer Graphics Forum*, 2003. 8

[36] Y. Peng, A. Ganesh, J. Wright, W. Xu, and Y. Ma. Rasl: Robust alignment by sparse and low-rank decomposition for linearly correlated images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(11):2233–2246, Nov. 2012. 6, 7

[37] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *CoRR*, abs/1706.02413, 2017. 8, 15

[38] I. Radosavovic, P. Dollr, R. Girshick, G. Gkioxari, and K. He. Data distillation: Towards omni-supervised learning. *arXiv preprint arXiv:1712.04440*, 2017. 2

[39] L. Rokach. Ensemble-based classifiers. *Artif. Intell. Rev.*, 33(1-2):1–39, Feb. 2010. 9

[40] M. Rubinstein, A. Joulin, J. Kopf, and C. Liu. Unsupervised joint object discovery and segmentation in internet images. In *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*, pages 1939–1946. IEEE Computer Society, 2013. 1

[41] M. Rubinstein, C. Liu, and W. T. Freeman. Joint inference in weakly-annotated image datasets via dense correspondence. *Int. J. Comput. Vision*, 119(1):23–45, Aug. 2016. 1

[42] R. M. Rustamov. Laplace-beltrami eigenfunctions for deformation invariant shape representation. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, SGP '07, pages 225–233, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association. 6

[43] Y. Shen, Q. Huang, N. Srebro, and S. Sanghavi. Normalized spectral map synchronization. In *Neural Information Processing Systems (NIPS)*, 2016. 1, 2

[44] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3d. *ACM Trans. Graph.*, 25(3):835–846, July 2006. 1

[45] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser. Semantic scene completion from a single depth image. *Proceedings of 30th IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 8

[46] R. E. Tarjan. Edge-disjoint spanning trees and depth-first search. *Acta Inf.*, 6(2):171–185, June 1976. 4

[47] F. Wang, Q. Huang, and L. Guibas. Image co-segmentation via consistent functional maps. In *In Proceedings of the 14th International Conference on Computer Vision (ICCV)*, 2013. 1, 5

[48] F. Wang, Q. Huang, M. Ovsjanikov, and L. J. Guibas. Unsupervised multi-class joint image segmentation. In *CVPR*, pages 3142–3149. IEEE Computer Society, 2014. 1

[49] L. Wang and A. Singer. Exact and stable recovery of rotations for robust synchronization. *CoRR*, abs/1211.2441, 2012. 2

[50] Y. Wang, S. Asafi, O. van Kaick, H. Zhang, D. Cohen-Or, and B. Chen. Active co-analysis of a set of shapes. *ACM Trans. Graph.*, 31(6):165:1–165:10, Nov. 2012. 6, 15

[51] Y. Yang and D. Ramanan. Articulated human detection with flexible mixtures of parts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(12):2878–2890, Dec. 2013. 7

[52] Z. Yi, H. Zhang, P. Tan, and M. Gong. Dualgan: Unsupervised dual learning for image-to-image translation. *CoRR*, abs/1704.02510, 2017. 1, 2

[53] Yuxin Chen and E. Candes. The projected power method: An efficient algorithm for joint alignment from pairwise differences. *https://arxiv.org/abs/1609.05820*, 2016. 2

[54] C. Zach, M. Klopschitz, and M. Pollefeys. Disambiguating visual relations using loop constraints. In *CVPR*, pages 1426–1433. IEEE Computer Society, 2010. 2, 6

[55] A. R. Zamir, A. Sax, W. B. Shen, L. J. Guibas, J. Malik, and S. Savarese. Taskonomy: Disentangling task transfer learning. *CoRR*, abs/1804.08328, 2018. 2

[56] T. Zhou, P. Krähenbühl, M. Aubry, Q. Huang, and A. A. Efros. Learning dense correspondence via 3d-guided cycle consistency. In *Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 2, 6, 7

[57] T. Zhou, Y. J. Lee, S. X. Yu, and A. A. Efros. Flowweb: Joint image set alignment by weaving consistent, pixel-wise correspondences. In *CVPR*, pages 1191–1200. IEEE Computer Society, 2015. 1, 2, 6, 7

[58] X. Zhou, M. Zhu, and K. Daniilidis. Multi-image matching via fast alternating minimization. In *ICCV*, pages 4032–4040, Santiago, Chile, 2015. IEEE Computer Society. 1, 2, 6

[59] J. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593, 2017. 1, 2

A. Proof of Theorems and Propositions

A.1. Proof of Proposition 1

To show that $\mathcal{F}$ is path-invariant, it suffices to prove that $f_p = f_q$ for every path pair $(p, q) \in \mathcal{G}_{\text{pair}}$. But by Definition 7, (p, q) is either in $\mathcal{G}_{\text{pair}}$ or can be induced from a finite number of operations with merge, stitch and/or cut. So if we can show that the output path pair in every round of operation keeps consistency on the map network $\mathcal{F}$, given the input path pairs are consistent, then all path pairs on $\mathcal{G}$ would be path-invariant by employing an induction proof. Next we achieve this goal by considering three operations respectively.

- **merge.** The merge operation takes as input two path pairs $(p, q), (p', q') \in \mathcal{G}_{\text{pair}}$ where $p' = r \sim p \sim r'$, i.e., p' is formed by stitching three sub-paths r, p and r' in order. By Definition 2, it is easy to see that

  $$f_{p'} = f_{r'} \circ f_p \circ f_r.$$

  But we are given that $\mathcal{F}$ is consistent on the input pairs, or equivalently,

  $$f_p = f_q, \quad f_{p'} = f_{q'}.$$

  Hence

  $$f_{q'} = f_{p'} = f_{r'} \circ f_q \circ f_r = f_{r \sim q \sim r'}.$$

  So $\mathcal{F}$ is also consistent on path pair $(r \sim q \sim r', q')$.

- **stitch.** The stitch operation takes as input two path pairs $(p, q), (p', q')$ where $p, q \in \mathcal{G}_{\text{path}}(u, v)$ and $p', q' \in \mathcal{G}_{\text{path}}(v, w)$. Since $\mathcal{F}$ is consistent on (p, q) and (p', q'), it follows immediately

  $$f_{p \sim p'} = f_{p'} \circ f_p = f_{q'} \circ f_q = f_{q \sim q'},$$

  which means $\mathcal{F}$ is also consistent on $(p \sim p', q \sim q')$.

- **cut.** The cut operation takes as input two path pairs $(\mathcal{C}_1, \varnothing)$ and $(\mathcal{C}_2, \varnothing)$, where $\mathcal{C}_1$ and $\mathcal{C}_2$ are two common vertices u, v and share a common intermediate path from v to u. The two cycles can be represented by $u \xrightarrow{p} v \xrightarrow{q} u$ and $u \xrightarrow{p'} v \xrightarrow{q} u$ where $p, p' \in \mathcal{G}_{\text{path}}(u, v)$ and $q \in \mathcal{G}_{\text{path}}(v, u)$. Since $\mathcal{F}$ is consistent on $(\mathcal{C}_1, \varnothing)$ and $(\mathcal{C}_2, \varnothing)$, we have

  $$f_{p \sim q} = f_q \circ f_p = I, \quad f_{p' \sim q} = f_q \circ f_{p'} = I.$$

  However, it is known that the inverse of some function must be unique, giving the following result

  $$f_{p'} = f_p,$$

  or in other words, $\mathcal{F}$ is consistent on path pair (p, p').

The consistency of $\mathcal{F}$ on the output pairs for all three operations given the consistency on their input pairs ensures our proposition. □

A.2. Proof of Theorem 3.1

The algorithm adds exactly $|\mathcal{E}|$ edges in total. And during each edge insertion, at most $|\mathcal{P}| \leq |\mathcal{V}|$ path pairs would be added to $\mathcal{B}_{dag}(\sigma)$, thus it follows immediately that $|\mathcal{B}_{dag}(\sigma)| \leq |\mathcal{V}||\mathcal{E}|$.

Next we show that $\mathcal{B}_{dag}(\sigma)$ indeed is a path-invariance basis for $\mathcal{G}$. To this end, we will verify that every path pair in $\mathcal{G}_{\text{pair}}$ can be induced from a subset of $\mathcal{B}_{dag}(\sigma)$ by operations, using a induction proof. In particular, we claim that at all time points, all path pairs in $\mathcal{G}_{cur}$ can be induced from $\mathcal{B}_{dag}(\sigma)$ by a series of operations. Initially, this inductive assumption holds trivially since $\mathcal{G}_{cur}$ is an empty set.

Suppose now we were processing an edge $(u, v) \in \mathcal{E}$ (so $(u, v) \notin \mathcal{G}_{cur}(\sigma)$ at this time point) and let $\mathcal{G}'_{cur} = \mathcal{G}_{cur} \cup \{(u, v)\}$. By inductive assumption, all path pairs in $\mathcal{G}_{cur}$ can be induced from $\mathcal{B}_{dag}(\sigma)$. After inserting (u, v) into $\mathcal{G}_{cur}$, it suffices to consider path pairs that contain edge (u, v) since all other path pairs have been guaranteed by inductive assumption. Let (p, q) be a path pair in $\mathcal{G}'_{cur}$ containing (u, v). Without loss of generality, suppose $p, q \in \mathcal{G}_{\text{path}}(w, v)$ and

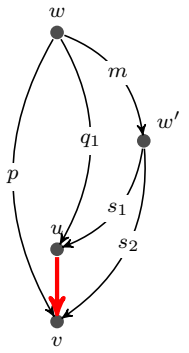


Figure 7: An Illustration for Path-Pair Generation

$$p = p_1 \sim (r, v),\ q = q_1 \sim (u, v).$$

If $r = u$, then (p, q) can be induced by stitching (p_1, q_1) and $((u, v), (u, v))$ where $(p_1, q_1) \in \mathcal{G}_{cur}$. We assume $r \neq u$, and then p would be a path from w to v in $\mathcal{G}_{cur}$ and q_1 would be a path from w to u in $\mathcal{G}_{cur}$.

Recall the definition of $\mathcal{P}$ and $\overline{\mathcal{P}}$. $w \in \mathcal{P}$ immediately follows. If $w \notin \overline{\mathcal{P}}$, then there exists $w' \neq w$ such that w can reach w' in $\mathcal{G}_{cur}$ and $w' \in \overline{\mathcal{P}}$ and denote such path as m. For convenience, we let $w' = w$ and $m = \varnothing$ when $w \in \overline{\mathcal{P}}$. Every vertex in $\overline{\mathcal{P}}$ corresponds to a path-invariance pair to be added to $\mathcal{B}_{dag}$ by our algorithm. Here we assume that it is $(s_1 \sim (u, v), s_2)$ for w' where $s_1 \in \mathcal{G}_{\text{path}}(w', u)$, $s_2 \in \mathcal{G}_{\text{path}}(w', v)$ and s_2 is within $\mathcal{G}_{cur}$.

By the property of DAG and the order of edge insertion, all paths from w to u in $\mathcal{G}$ are also in $\mathcal{G}_{cur}$ since $(u, v) \in \mathcal{E}$. Thus $(q_1, m \sim s_1)$ can be induced from $\mathcal{B}_{dag}(\sigma)$ by inductive assumption. Similarly, as s_2 is within $\mathcal{G}_{cur}$, $(p, m \sim s_2)$ is also a path-invariance pair, which can be induced from $\mathcal{B}_{dag}(\sigma)$. Next we give the operation steps to build (p, q):

$$(m, m) + (s_1 \sim (u, v), s_2) \xrightarrow{stitch} (m \sim s_1 \sim (u, v), m \sim s_2) \quad (6)$$

$$(m \sim s_1 \sim (u, v), m \sim s_2) + (q_1, m \sim s_1) \xrightarrow{merge} (q_1 \sim (u, v), m \sim s_2) \quad (7)$$

$$(q_1 \sim (u, v), m \sim s_2) + (p, m \sim s_2) \xrightarrow{merge} (p, q) \quad (8)$$

For the last step, notice that $q = q_1 \sim (u, v)$ and (p, q) is equivalent to (q, p). Thus all path pairs in $\mathcal{G}'_{cur}$ can be induced by path pairs in $\mathcal{B}_{dag}(\sigma)$ with a series of operations, which completes our proof by induction. □

A.3. Proof of Proposition 2

Before proving Proposition 2, we first introduce some well-known terms for depth-first search. There are two time stamps $d[v]$ and $f[v]$ for each vertex v, where d is defined as the time point when it visits v for the first time and f as the time point when it finishes visiting v. Some edge (u, v) in $\mathcal{E}$ can be classified into one of four disjoint types as follows:

- **Tree Edge:** v is visited for the first time as we traverse the edge (u, v). In this case (u, v) will be added into the resulting DFS spanning tree. For tree edge we have
$$d[u] < d[v], \quad f[u] > f[v].$$
- **Back Edge:** v is visited and is an ancestor of u in the current spanning tree. For back edge we have
$$d[u] > d[v], \quad f[u] < f[v].$$
- **Forward Edge:** u is visited and is an ancestor of v in the current spanning tree. For forward edge we have
$$d[u] < d[v], \quad f[u] > f[v].$$
- **Cross Edge:** v is visited and is neither an ancestor nor descendant of u in the current spanning tree. For cross edge we have
$$d[u] > d[v], \quad f[u] > f[v].$$

Using these definitions, we prove that:

Any cycle $\mathcal{C}$ in $\mathcal{G}$ have a vertex u in $\mathcal{C}$ such that all other vertices are located within the sub-tree rooted at u, i.e., are the descendants of u in $\mathcal{T}$.

Let

$$\mathcal{C} : u_1 \to u_2 \to \cdots \to u_k \to u_1.$$

Without loss of generality, u_1 is assumed to be the one with smallest d among all $\{u_i\}$. If not all u_i are descendants of u_1, we choose u_t to be the one with smallest t, which

means u_{t-1} is a descendant of u_1 but u_t is not. Obviously (u_{t-1}, u_t) cannot be a tree edge or forward edge, which causes u_t to be a descendant of u_{t-1} and also a descendant of u_1. If (u_{t-1}, u_t) is a back edge, then u_t is not a descendant of u_1 if and only if $u_{t-1} = u_1$ since there is in fact unique back path in the spanning tree $\mathcal{T}$. But $u_{t-1} = u_1$ means u_t is a parent of u_1, and thus there exists a smaller d than u_1, which results in a contradiction. Also (u_{t-1}, u_t) cannot be a cross edge. In fact, since u_{t-1} is a descendant of u_1, we have $f[u_{t-1}] < f[u_1]$. Together with $f[u_t] < f[u_{t-1}]$ from cross edge property, we have $f[u_t] < f[u_1]$. But u_t is not a descendant or ancestor of u_1, which means the sub-tree rooted at u_1 is disjoint from the sub-tree rooted at u_t, so intervals $[d[u_1], f[u_1]]$ and $[d[u_t], f[u_t]]$ must be disjoint by the property of depth-first search. As thus $f[u_t] < f[u_1]$ implies $d[u_t] < d[u_1]$, which contradicts the assumption that $d[u_1]$ is smallest among u_i. Hence all u_i are descendants of u_1.

Now come back to the original proposition. Continue using the notation $\mathcal{C}$ defined above. In addition we define $\mathcal{C}_i$ as the sub-path from u_1 to u_i, i.e.,

$$\mathcal{C}_i : u_1 \to u_2 \to \cdots \to u_i.$$

We will show $\mathcal{C}$ can be induced from $\mathcal{B}$ by a finite number of operations with merge, stitch and cut. Above all, we have assumed the property of path-invariance on $\mathcal{T}$ by Theorem 3.1. Given u_1 is the common ancestor of all u_i, we inductively prove the following statement:

The path $u_1 \to u_2 \to \cdots \to u_t$ $(t \le k)$ is equivalent to the tree path from u_1 to u_t. Here tree path means a path in which all edges are in the spanning tree $\mathcal{T}$.

The base case is trivial. Now suppose $u_1 \to u_2 \to \cdots \to u_t$ $(t < k)$ is equivalent to tree path P from u_1 to u_t and we continue to check $u_1 \to u_2 \to \cdots \to u_{t+1}$.

- If (u_t, u_{t+1}) is a tree edge, then $P \sim (u_t, u_{t+1})$ is still a tree path and a stitch operation on path pair $(\mathcal{C}_t, P)$ and $((u_t, u_{t+1}), (u_t, u_{t+1}))$ gives the equivalency that we want.
- If (u_t, u_{t+1}) is a forward edge, then there exists a tree path P_1 from u_t to u_{t+1}. By path-invariance on $\mathcal{G}_{dag}$, we can stitch two path-invariance pair $(\mathcal{C}_t, P)$ and $((u_t, u_{t+1}), P_1)$ to obtain the desired equivalency.
- If (u_t, u_{t+1}) is a back edge, then there exists a tree path P_1 from u_{t+1} to u_t. In addition by our construction the cycle $P_1 \sim (u_t, u_{t+1})$ has been added into our basis set $\mathcal{B}$. Denote the tree path from u_1 to u_{t+1} as P_2, then stitching (P_2, P_2) and $(P_1 \sim (u_t, u_{t+1}), \varnothing)$ gives $(P_2 \sim P_1 \sim (u_t, u_{t+1}), P_2)$. On the other hand, by inductive assumption we have path-invariance pair $(P_2 \sim P_1, \mathcal{C}_t)$ since $P_2 \sim P_1$ is just the tree path from u_1 to u_t. Thus by merging $(P_2 \sim P_1, \mathcal{C}_t)$ and $(P_2 \sim P_1 \sim (u_t, u_{t+1}), P_2)$ we obtain the path pair $(\mathcal{C}_t \sim (u_t, u_{t+1}), P_2)$, or equivalently, $(\mathcal{C}_{t+1}, P_2)$.
- If (u_t, u_{t+1}) is a cross edge, then (u_t, u_{t+1}) has been included in $\mathcal{G}_{dag}$. Denote by P_1 the tree path from u_1 to u_t. In this way all $P_1 \sim (u_t, u_{t+1})$ would be equivalent to another tree path P_2 from u_1 to u_{t+1} since all edges involved here are within $\mathcal{G}_{dag}$ which maintains all possible path-invariance pairs. By merging path pairs $(P_1 \sim (u_t, u_{t+1}), P_2)$ and $(P_1, \mathcal{C}_t)$ we obtain path pair $(\mathcal{C}_t \sim (u_t, u_{t+1}), P_2)$, or $(\mathcal{C}_{t+1}, P_2)$, which is exactly we want to verify.

As thus we finished our inductive proof. In particular, the path (also a cycle) $u_1 \to \cdots \to u_k \to u_1$ is equivalent to $\varnothing$, or more precisely, the path pair $(\mathcal{C}, \varnothing)$ can be induced from $\mathcal{B}$ by a finite number of merge and stitch operations.

To complete our proof, we need to show that all path pairs in $\mathcal{G}$ instead of just $\mathcal{G}_{dag}$ can be induced from $\mathcal{B}$. This is relatively easy. Consider two path P_1 and P_2 both from u to v. Since $\mathcal{G}$ is strongly connected, there must exist some path P_3 from v to u. The cut operation on $P_1 \sim P_3$ and $P_2 \sim P_3$ for the common vertices u and v immediately gives the path pair (P_1, P_2). □

A.4. Proof of Theorem 3.2

To prove this theorem, we first prove the following lemma:

Lemma *Suppose $\mathcal{G}_i$ and $\mathcal{G}_j$ are two strongly connected components in $\mathcal{G}$ with $(\mathcal{G}_i, \mathcal{G}_j) \in \mathcal{G}_{dag}$. Given any vertices $u, u' \in \mathcal{G}_i$ and $v, v' \in \mathcal{G}_j$ with $(u, v), (u', v') \in \mathcal{E}_{ij}$, and paths $p \in \mathcal{G}_{\text{path}}(u, u')$, $p' \in \mathcal{G}_{\text{path}}(v, v')$, we claim that $p \sim (u', v')$ is equivalent to $(u, v) \sim p'$ under $\mathcal{B}$.*

In fact since $\mathcal{B}$ ensures equivalence for all path pairs inside the same SCC, the specific p, p' does not matter. We only care about the starting and ending points when everything happens inside a single SCC. So in the following proof we will use $P(x, y)$ to denote some path from x to y inside the single SCC but not mentioning the intermediate vertices. Recall that we built a (undirected) spanning tree $\mathcal{T}$ on $\mathcal{E}_{ij}^2$. Thus we have an edge sequence

$$(u_1, v_1), (u_2, v_2), \ldots, (u_k, v_k)$$

where $u_1 = u$, $v_1 = v$, $u_k = u'$, $v_k = v'$, and edge pair

$$((u_l, v_l), (u_{l+1}, v_{l+1}))$$

are in $\mathcal{T}$ for all $l = 1, \ldots, k-1$. Next we inductively prove that $P(u_1, u_t) \sim (u_t, v_t)$ is equivalent to $(u_1, v_1) \sim P(v_1, v_t)$ for $t = 1, \ldots, k$. The base case where $t = 1$ is trivial. Given the correctness for t, consider $t + 1$. It is known that $(u_t, v_t) \sim P(v_t, v_{t+1})$ is equivalent to $P(u_t, u_{t+1}) \sim (u_{t+1}, v_{t+1})$ by the construction of $\mathcal{T}$ and $P(u_1, u_t) \sim (u_t, v_t)$ is equivalent to $(u_1, v_1) \sim P(v_1, v_t)$ by inductive assumption. By successively applying two merge operations on path

$$(u_1, v_1) \sim P(v_1, v_t) \sim P(v_t, v_{t+1})$$

Figure 8: Visual comparison between our approach and state-of-the-art approaches. This figure is best viewed in color, zoomed in.

we obtain the equivalent path

$$P(u_1, u_t) \sim P(u_t, u_{t+1}) \sim (u_{t+1}, v_{t+1}).$$

But it is straightforward that $P(u_1, u_t) \sim P(u_t, u_{t+1})$ is equivalent to $P(u_1, u_{t+1})$ under $\mathcal{B}$ since u_1, u_t, u_{t+1} are in the same SCC. Similarly, $P(v_1, v_t) \sim P(v_t, v_{t+1})$ is equivalent to $P(v_1, v_{t+1})$. Thus finally we obtain the equivalency on $P(u_1, u_{t+1}) \sim (u_{t+1}, v_{t+1})$ and $(u_1, v_1) \sim P(v_1, v_{t+1})$, which completes our inductive proof and the lemma immediately follows.

Come back to the original theorem. With notation $P(x, y)$, we can express an arbitrary path p in $\mathcal{G}$ from u to v as

$$\begin{aligned} p : P(u_1, v_1) \sim (v_1, u_2) \sim P(u_2, v_2) \sim \\ \cdots \sim (v_{k-1}, u_k) \sim P(u_k, v_k) \end{aligned}$$

where $u_1 = u$, $v_k = v$, and u_i, v_i are in the same SCC $\mathcal{G}_{b_i}$. Similarly write another path p' from u to v this way:

$$\begin{aligned} p' : P(u'_1, v'_1) \sim (v'_1, u'_2) \sim P(u'_2, v'_2) \sim \\ \cdots \sim (v'_{k-1}, u'_k) \sim P(u'_k, v'_k) \end{aligned}$$

where $u'_1 = u$, $v'_k = v$, and u_i, v_i are in the same SCC $\mathcal{G}_{b'_i}$ with obvious constraints $b_1 = b'_1$ and $b_k = b'_k$. As we extend $\mathcal{B}_{dag}$ that maintains the equivalency on all possible pairs in $\mathcal{G}_{dag}$ to $\mathcal{G}$, there would be a path pair

$$\begin{aligned} q : P(\alpha_1, \beta_1) \sim (\beta_1, \alpha_2) \sim P(\alpha_2, \beta_2) \sim \\ \cdots \sim (\beta_{k-1}, \alpha_k) \sim P(\alpha_k, \beta_k) \\ q' : P(\alpha'_1, \beta'_1) \sim (\beta'_1, \alpha'_2) \sim P(\alpha'_2, \beta'_2) \sim \\ \cdots \sim (\beta'_{k-1}, \alpha'_k) \sim P(\alpha'_k, \beta'_k) \end{aligned}$$

in the extended $\mathcal{B}_{dag}$ where $\alpha_1 = \alpha'_1$, $\beta_k = \beta'_k$, and α_i, β_i are in the same SCC $\mathcal{G}_{b_i}$ while α'_i, β'_i in $\mathcal{G}_{b'_i}$. Thus it suffices to prove that p is equivalent to $P(u_1, \alpha_1) \sim q \sim P(\beta_k, v_k)$ while p' equivalent to $P(u'_1, \alpha'_1) \sim q' \sim P(\beta'_k, v'_k)$. (Recall that $u'_1 = u_1$, etc.) Since the proofs for them are essentially identical, we only consider p.

In fact, $P(u_1, \alpha_1) \sim q \sim P(\beta_k, v_k)$ can be equivalently expressed as

$$\begin{aligned} P(u_1, v_1) \sim P(v_1, \beta_1) \sim (\beta_1, \alpha_2) \sim P(\alpha_2, u_2) \\ \sim P(u_2, v_2) \sim P(v_2, \beta_2) \sim \dots \\ \sim P(\beta_{k-1}, \alpha_k) \sim (\alpha_k, u_k) \sim P(u_k, v_k). \end{aligned}$$

In other words, we split $P(\alpha_i, \beta_i)$ into $P(\alpha_i, u_i) \sim P(u_i, v_i) \sim P(v_i, \beta_i)$ for $i = 2, \dots, k-1$. However, our lemma just states that

$$P(v_i, \beta_i) \sim (\beta_i, \alpha_{i+1})$$

is equivalent to

$$(v_i, u_{i+1}) \sim P(u_{i+1}, \alpha_{i+1}).$$

Thus by series of merge operations, $P(u_1, \alpha_1) \sim q \sim P(\beta_k, v_k)$ can be shown to be equivalent to

$$\begin{aligned} P(u_1, v_1) \sim (v_1, u_2) \sim P(u_2, \alpha_2) \sim P(\alpha_2, u_2) \\ \sim P(u_2, v_2) \sim P(v_2, u_3) \sim \dots \\ \sim P(u_k, \alpha_k) \sim P(\alpha_k, u_k) \sim P(u_k, v_k), \end{aligned}$$

which is clearly p by cancelling all consecutive P's. □

A.5. Proof of Proposition 3

First note that in fact the bound $|\mathcal{V}||\mathcal{E}|$ theorem 3.1 can be improved to $(|\mathcal{V}|-1)|\mathcal{E}|$ since $|\mathcal{P}| \leq |\mathcal{V}| - 1$ all time.

In this way the size of $\mathcal{B}_i$ is bounded by $|\mathcal{E}(\mathcal{B}_i)|(|\mathcal{V}(\mathcal{B}_i)| - 1)$. Suppose there are k strongly connected components in $\mathcal{G}$ and c edges across different SCCs. Then there are at most c edges in $\bigcup_{i,j} \mathcal{B}_{ij}$ since the edge number of a spanning tree is less than that of vertices by 1. Notice c is also the edge number of contracted graph $\mathcal{G}_{dag}$. Hence for the $\mathcal{G}_{dag}$, there are would be at most $(k-1) \times c$ items in $\mathcal{B}_{dag}$. Also observe that each SCC can have at most $|\mathcal{V}| - k + 1$ vertices when there are k SCCs. So the size of $\mathcal{B}$ would be bounded by

$$\begin{aligned} &(k-1)c + c + (|\mathcal{V}| - k)|\mathcal{E}| \\ \leq &k|\mathcal{E}| + (|\mathcal{V}| - k)|\mathcal{E}| \\ = &|\mathcal{V}||\mathcal{E}| \end{aligned}$$

□

B. Additional Details of Joint Dense Image Map

B.1. Training Details

We applied ADAM [27] to solve the following optimization problem for predicting dense image correspondences.

$$\min_{\theta} \sum_{(i,j)\in\mathcal{E}} \|f_{ij}^{\theta} - f_{ij}^{in}\|_1 + \lambda \sum_{(p,q)\in\mathcal{B}} \|f_p^{\theta} - f_q^{\theta}\|_{\mathcal{F}}^2 \quad (9)$$

We initialize f^{θ} by directly fitting it to the input image flows between pairs of images. We then impose the path-invariance regularization term to improve the network flow.

B.2. More Qualitative Evaluations

Figure 8 and Figure 9 provide more qualitative evaluations of our approach on the PASCAL Rigid categories. Besides the two categories shown in the main paper (Car and Aeroplane), we pick one example from the remaining 10 rigid categories. Note that our approach is consistently better than all baseline approaches.

C. Additional Details of 3D Semantic Scene Segmentation

C.1. Network Architecture and Training Details

For point cloud semantic segmentation network, we follow the same configuration from PointNet++ [37]. For voxel semantic segmentation network, we use the same network architecture proposed in 3D U-Net[9]. To generate training data from ScanNet scenes, following PointNet++ [37], we sample 1.5m by 1.5m by 3m cubes from the initial scenes. We sample such training cubes on the fly and randomly rotate each sample along the up-right axis. During test time, we split the test scene into smaller cubes first, and then merge label prediction in all the cubes from the same scene. Note that this is done for the prediction using each 3D representation in isolation.

We applied ADAM [27] to solve the optimization problem for predicting semantic labels in 3D scenes. We first initialize network parameters using the pre-trained weight on labeled data, and then impose the path-invariance regularization term to improve the network performance.

C.2. More Quantitative Evaluations

Table 2 shows per-class semantic voxel label prediction accuracy on ScanNet [12] test scenes. Compared to baseline methods, our approach shows consistently better performance compared to using 8% labeled data, and competitive results compared to using 30% and 100% labeled data, especially on frequently appeared classes, such as floor, wall, chair, sofa, and etc.

C.3. More Qualitative Evaluations

Figure 10 presents more qualitative comparisons between our approach and baselines. Consistently, using 8% labeled data and 92% unlabeled data, our approach achieved competing performance as using 30% to 100% labeled data when trained on each individual representation, and better performance as using 8% labeled data.

D. Additional Details of Joint Shape Matching

D.1. Training Details

We applied ADAM [27] to solve the following optimization problem:

$$\sum_{(i,j)\in\mathcal{E}} \|X_{ij} - X_{ij}^{in}\|_1 + \lambda \sum_{(p,q)\in\mathcal{B}} \|X_p - X_q\|_{\mathcal{F}}^2 \quad (10)$$

Initially, we set $X_{ij} = X_{ij}^{in}$. We also tried reweighted non-linear least squares and used Gauss-Newton optimization to solve the induced non-linear least square problem (we used conjugate gradient to solve the induced linear system) . We found that the optimal solutions of both approaches are similar, suggesting both of them reached a strong local minimum. Computationally, we find the ADAM optimizer to be more efficient.

D.2. Annotated Feature Points

D.2.1 SHREC07

We used annotated feature points provided by [26]. The number of key points per category range from 11 (e.g. Plane) to 36 (Human).

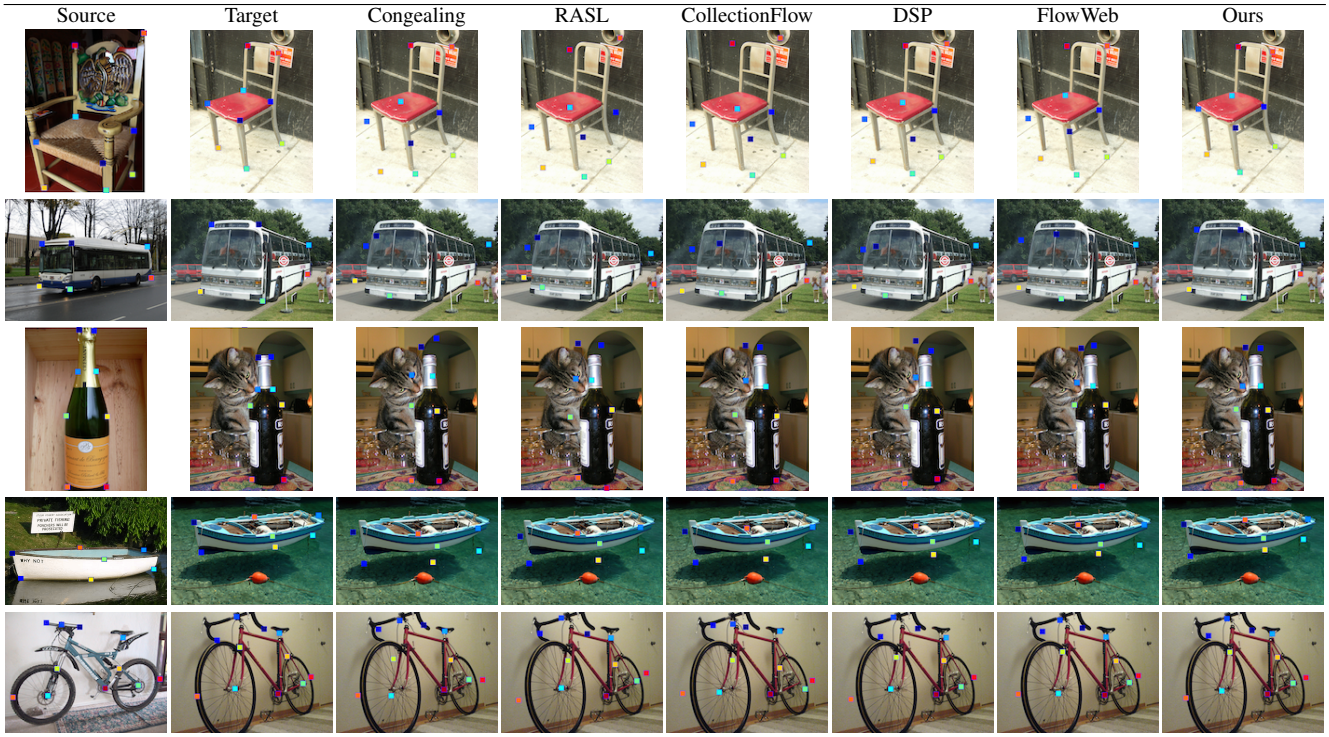


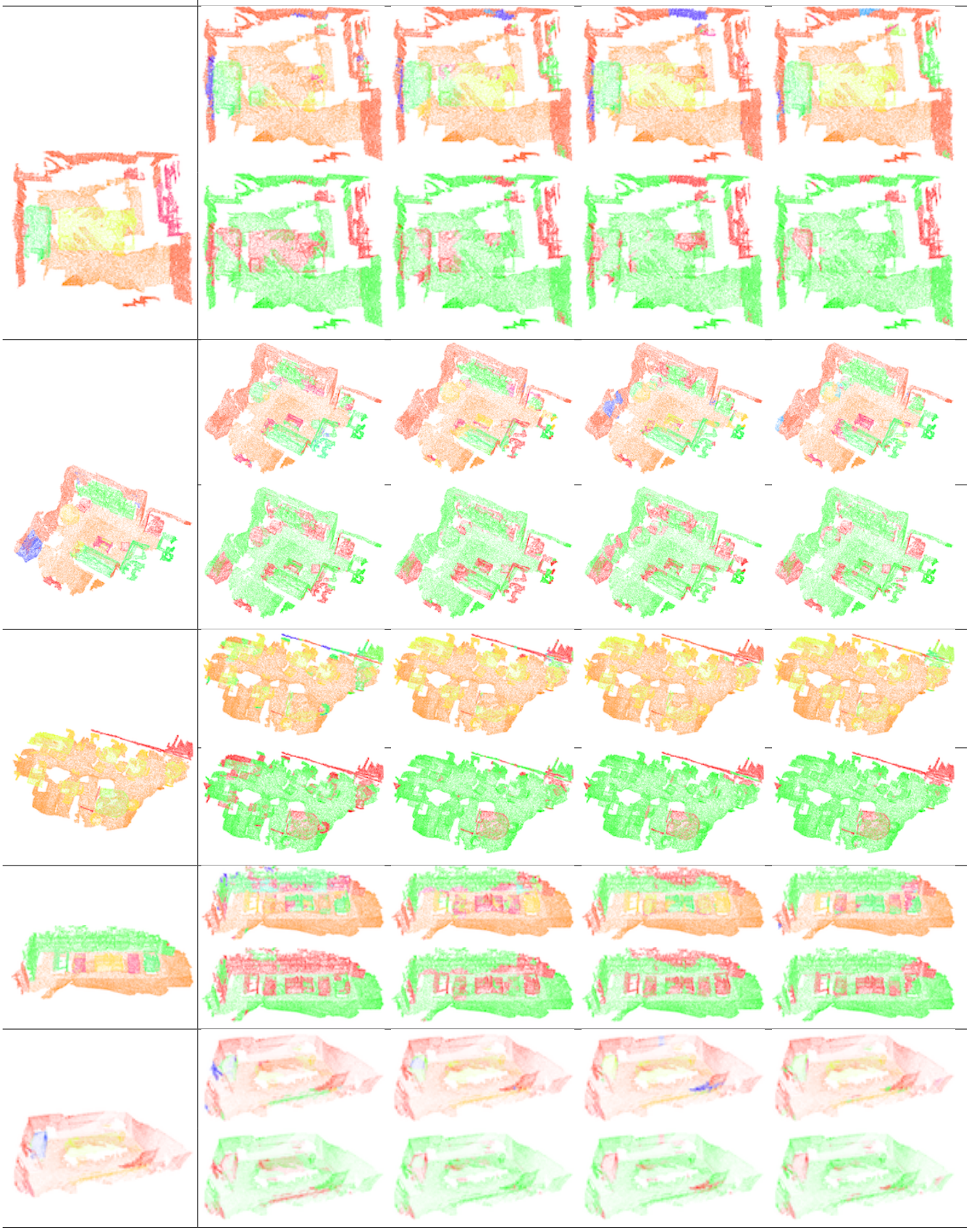
Figure 9: Visual comparison between our approach and state-of-the-art approaches. This figure is best viewed in color, zoomed in.

	Floor	Wall	Chair	Sofa	Table	Door	Cabinet	Bed	Desk	Toilet	Sink	Window	Picture	BookSh	Curtain	ShowerC	Counter	Fridge	Bathtub	OtherF	Total
Weight	35.7	38.8	3.8	2.5	3.3	2.2	2.4	2.0	1.7	0.2	0.2	0.4	0.2	1.6	0.7	0.04	0.6	0.3	0.2	2.9	-
PCI	90.9	98.1	58.4	45.4	40.2	47.4	36.4	62.8	21.8	35.4	32.0	16.7	21.5	0.0	0.0	1.3	0.0	0.0	19.7	9.6	79.2
	93.3	98.4	70.3	54.8	50.0	49.2	80.9	87.1	18.4	83.7	58.9	8.4	0.2	1.0	1.8	2.9	3.7	0.0	13.0	5.7	82.3
	88.0	97.8	76.3	62.7	19.9	63.5	65.5	59.7	52.5	63.9	76.2	17.4	27.1	17.0	12.2	56.1	0.0	0.0	25.7	22.0	80.8
	90.8	98.2	78.0	67.5	42.8	74.8	79.6	79.8	58.2	78.0	82.1	53.1	42.3	12.1	28.2	70.0	52.7	0.0	37.3	18.7	84.2
PCII	91.5	97.2	49.4	32.2	32.4	44.3	30.8	70.1	24.9	45.0	35.0	29.2	23.9	0.0	10.6	1.1	0.0	0.0	18.0	10.0	78.3
	94.9	98.4	65.0	58.1	48.0	41.7	65.4	89.6	31.2	81.0	62.9	4.6	4.6	0.0	0.4	3.7	0.0	0.0	17.5	4.5	82.5
	90.8	98.5	74.4	54.6	34.4	49.3	46.7	77.3	39.3	74.8	71.9	22.8	35.6	0.0	0.0	24.8	0.0	0.0	25.4	11.7	81.9
	92.8	98.0	86.4	64.2	29.8	55.0	59.2	75.3	37.6	86.5	67.6	9.3	25.3	23.5	19.0	46.6	43.1	0.0	25.0	13.7	83.3
PCIII	92.7	96.7	73.3	52.9	16.7	36.4	1.3	55.7	12.1	27.0	27.1	16.6	11.5	0.0	0.2	8.9	0.0	0.0	15.0	1.6	78.4
	93.7	98.1	71.4	58.9	50.0	54.4	59.9	74.8	30.6	82.8	65.1	10.6	1.6	1.4	0.8	21.5	0.0	0.0	20.3	8.7	82.3
	90.8	98.5	74.4	54.6	34.4	49.3	46.7	77.3	39.3	74.8	71.9	22.8	35.6	0.0	0.0	24.8	0.0	0.0	25.4	11.7	81.2
	90.4	97.6	76.1	65.0	45.5	80.6	70.9	75.3	32.4	82.0	73.9	48.0	49.8	13.5	16.9	64.4	46.7	0.0	42.0	13.0	83.4
VOLI	93.4	97.3	71.9	68.0	16.2	0.2	0.0	58.1	34.3	25.1	3.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	18.3	8.6	78.7
	93.5	97.6	70.7	61.2	55.7	39.1	55.0	76.7	11.5	81.3	68.8	0.3	2.3	2.2	0.0	2.0	0.0	0.0	16.8	10.2	81.6
	94.0	97.6	68.0	68.2	16.7	41.2	0.0	75.1	0.0	70.2	30.4	0.0	0.0	0.4	0.0	0.0	0.0	0.0	24.9	6.9	80.3
	92.5	97.5	74.2	67.2	25.0	55.0	59.5	62.9	0.0	85.4	0.0	3.9	38.5	0.4	0.0	0.0	42.5	0.0	37.8	14.2	81.9
VOLII	94.8	97.5	56.0	0.0	42.3	19.8	28.3	57.3	9.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	13.5	5.1	77.4
	92.8	97.7	69.6	0.0	53.8	31.6	66.4	68.2	11.4	77.3	0.0	0.0	0.0	0.0	1.1	0.0	0.0	0.0	19.8	9.8	79.0
	92.5	98.1	62.4	54.4	15.3	50.0	0.0	59.1	0.0	74.5	61.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	39.0	9.3	79.5
	91.0	96.9	68.4	60.5	31.4	59.1	70.0	81.2	0.0	86.3	0.0	11.1	0.0	0.0	1.4	0.0	0.0	0.0	50.1	15.1	81.5

Table 2: Per-class semantic voxel label prediction accuracy on ScanNet test scenes. All numbers are in percentages. The first row indicates the percentages of each class in all test scenes, and then for 4 rows in each representation, we show the per-class prediction accuracy in 4 configurations: 8% Label, 8% Label + 92%Unlabel, 30% Label and 100% Label. (BookSh, ShowerC and OtherF are short Bookshelf, Shower Curtain and Other furniture, respectively.)

D.2.2 ShapeCoSeg

Note that the models in ShapeCoSeg [50] are originally associated with annotations of semantic segments. Such annotations, however, are not ideal for establishing dense correspondences. To address this issue, we employed AMT to annotate semantic feature correspondences across the entire dataset. Note that in some cases, the feature correspondences are not purely based on 1-1 correspondences (e.g., multiple handles). When performing experimental evaluation, we evaluate the geodesic error to the closest feature point of the same type for experimental evaluation.


Ground Truth
8% Label
30% Label
100% Label
8% Label + 92%Unlabel

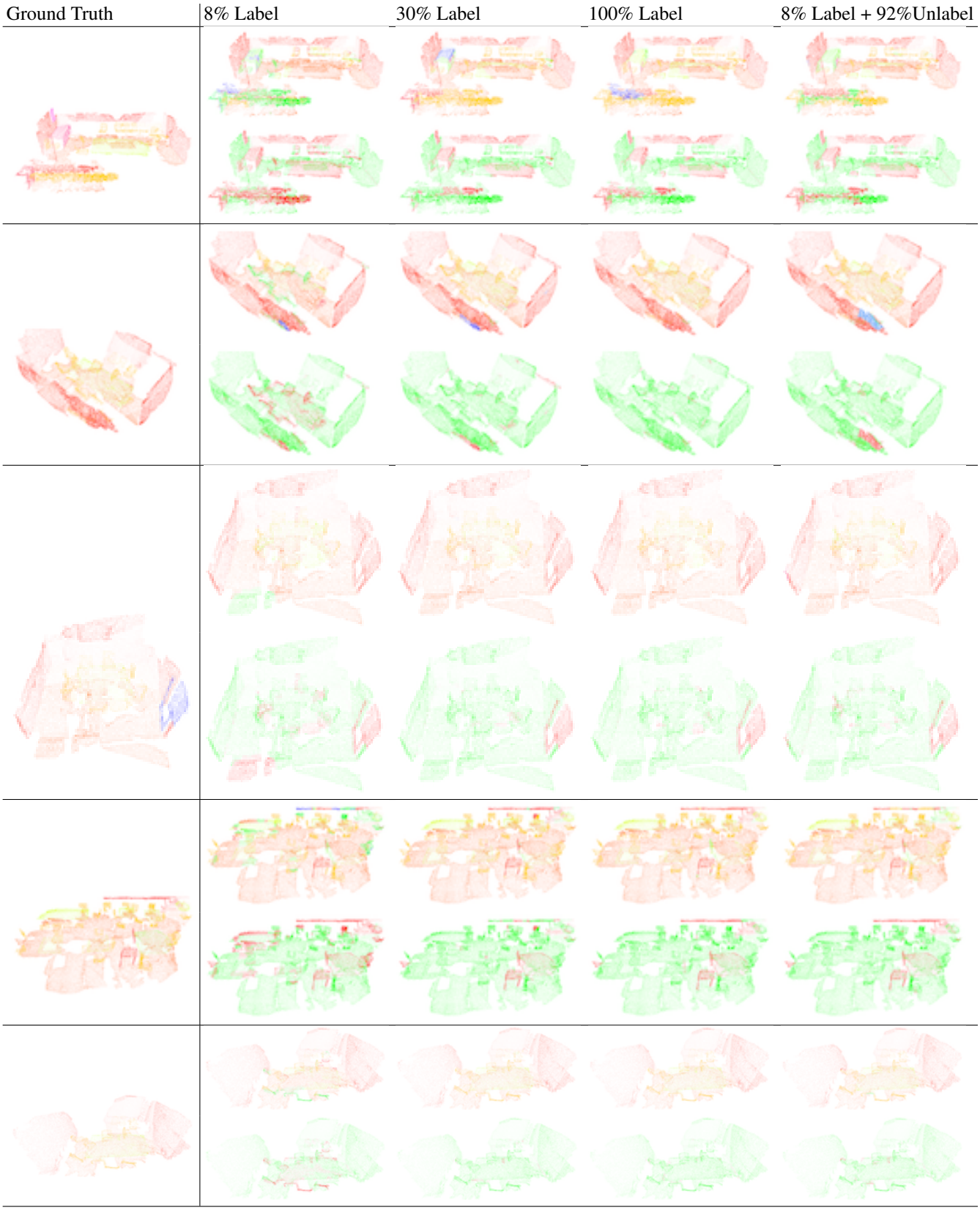


Figure 10: Qualitative comparisons of 3D semantic segmentation results on ScanNet [12]. Each row represents one testing instance, where ground truth and top sub-row show prediction for 21 classes and bottom sub-row only shows correctly labeled points. (Green indicates correct predictions, while red indicates false predictions.) This figure is best viewed in color, zoomed in.