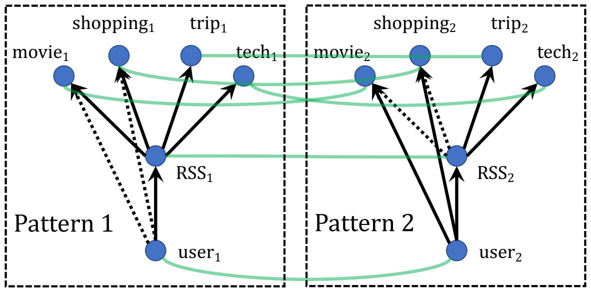


Figure 4. Schematic example showing necessity of latent topology. Solid lines correspond to "following" in real world. Dashed lines refer to latent connection. Green curved lines indicate matching.

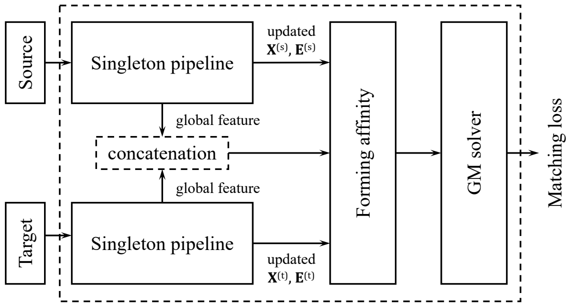


Figure 5. Holistic pipeline of DLGM consisting of two singleton pipelines.

A. Appendix

A.1. Necessity of latent topology

In some cases, graph topology/edge can have hard and explicit interpretation. However, in the context of learning, we anticipate our model can accommodate higher learning capacity and more generic scenarios. As such, we have emphasized our scenario in the 2nd footnote in page 1, where *topology construction is necessary* for matching. Even for some cases where hard and meaningful connection exists (e.g. social networks referred to R2), latent topology can still help to match. See Fig. 4 for an schematic example in social networks. We see the topology for both patterns are not identical (e.g. a direct "following" from $user_2$ to $movie_2$ exists in Pattern 2, but $user_1$ reads $movie_1$ via intermediate node RSS_1), which brings about misleading to match $movies$ and $users$ across graphs. However, if we are able to infer the latent link (dashed lines), we can make the two patterns topologically isomorphic and thus reduce the difficulty/ambiguity for matching. In general, *latent topology may not correspond to any exact meaning in real world, but its existence can potentially enhance matching.*

A.2. Holistic pipeline

We show the holistic pipeline of our framework in Fig. 5 consisting of two "singleton pipelines" (see introduction part of Sec. 3 for more details). In general, the holistic pipeline follows the convention in a series of deep graph matching methods by utilizing an identical singleton pipeline to extract features, then exploits the produced features to perform matching (Yu et al., 2020; Wang et al., 2019; Fey et al., 2020; Rolínek et al., 2020). Except for the topology module N_G, all others parts of our network are the same as those in Rolínek et al. (2020).

A.3. SplineCNN

SplineCNN is a method to perform graph-based representation learning via convolution operators defined based on B-splines (Fey et al., 2018). The initial input to SplineCNN is $\mathcal{G} = \{\mathbf{X}, \mathbf{E}, \mathbf{A}\}$, where $\mathbf{X} \in \mathcal{G}^{n\times d_1}$ and $\mathbf{A} \in \{0,1\}^{n\times n}$ indicate node features and topology, respectively (same as in Sec. 3.1). $\mathbf{E} \in [0,1]^{n\times n\times d_2}$ is so-called pseudo-coordinates and can be viewed as $n^2 \times d_2$-dimensional edge features for a fully connected graph (in case $m = n^2$, see Sec. 3.1). Let normalized edge feature $\mathbf{e}(i,j) = \mathbf{E}_{i,j,:} \in [0,1]^{d_2}$ if a directed edge (i,j) exists ($\mathbf{A}_{i,j} = 1$), and $\mathbf{0}$ otherwise ($\mathbf{A}_{i,j} = 0$). Note topology $\mathbf{A}$ fully carries the information of $\mathcal{N}(i)$ which defines the neighborhood of node i. During the learning, $\mathbf{X}$ and $\mathbf{E}$ will be updated while topology $\mathbf{A}$ will not. Therefore SplineCNN is a geometric graph embedding method without adjusting the latent graph topology.

B-spline is employed as basic kernel in SplineCNN, where a basis function has only support on a specific real-valued interval (Piegl & Tiller, 2012). Let $((N^q_{1,i})_{1\le i\le k_1}, ..., (N^q_{d,i})_{1\le i\le k_{d_2}})$ be d_2 B-spline bases with degree q. The kernel size is defined in $\mathbf{k} = (k_1, ..., k_{d_2})$. In SplineCNN, the continuous kernel function $g_l : [a_1, b_1] \times ... \times [a_{d_2}, b_{d_2}] \to \mathcal{G}$ is defined as:

$$g_l(\mathbf{e}) = \sum_{\mathbf{p}\in\mathcal{P}} w_{\mathbf{p},l} \cdot \mathbf{B}_{\mathbf{p}}(\mathbf{e}) \tag{21}$$

where $\mathcal{P} = (N^q_{1,i})_i \times ... \times (N^q_{d,i})_i$ is the B-spline bases (Piegl & Tiller, 2012) and $w_{\mathbf{p},l}$ is the trainable parameter corresponding to the lth node feature in $\mathbf{X}$, with $\mathbf{B}_{\mathbf{p}}$ being the product of the basis functions in $\mathbf{P}$:

$$\mathbf{B}_{\mathbf{p}} = \prod_{i=1}^{d} N^q_{i,p_i}(e_i) \tag{22}$$

where $\mathbf{e}$ is the pseudo-coordinate in $\mathbf{E}$. Then, given the kernel function $\mathbf{g} = (g_1, ..., g_{d_1})$ and the node feature $\mathbf{X} \in \mathcal{G}^{n\times d_1}$, one layer of the convolution at node i in SplineCNN reads (same as Eq. (3)):

$$(\mathbf{x} * \mathbf{g})(i) = \frac{1}{|\mathcal{N}(i)|} \sum_{l=1}^{d_1} \sum_{j\in\mathcal{N}(i)} x_l(j) \cdot g_l(\mathbf{e}(i,j)) \tag{23}$$

where $x_l(j)$ indicates the convolved node feature value of node j at lth dimension. This formulation can be tensorized into Eq. (4) with explicit topology matrix $\mathbf{A}$. In this sense, we can back-propagate the gradient of $\mathbf{A}$. Reader are referred to Fey et al. (2018) for more comprehensive understanding of this method.

A.4. Derivation of DLGM-D

We give more details of the optimization on DLGM-D in this section. This part also interprets some basic formulation conversion (e.g. from Eq. (2) to its Bayesian form). First, we assume there is no latent topology $\underline{\mathbf{A}}^{(s)}$ and $\underline{\mathbf{A}}^{(s)}$ at the current stage. In this case, the objective of GM is simply:

$$\max \prod_k P_\theta\left(\mathbf{Z}_k | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right) \tag{24}$$

where P_θ measures the probability of a matching $\mathbf{Z}_k$ given graph pair $\mathcal{G}_k^{(s)}$ and $\mathcal{G}_k^{(t)}$. If we impose the latent topology $\underline{\mathbf{A}}^{(s)}$ and $\underline{\mathbf{A}}^{(t)}$, as well as some *distribution* over them, then Eq. (24) can be equivalently expressed as:

$$\begin{aligned}&\max \prod_k P_\theta\left(\mathbf{Z}_k | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right)\\ =&\max \prod_k \int_{\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)}} P_\theta\left(\mathbf{Z}_k, \underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)} | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right)\end{aligned} \tag{25}$$

where $P_\theta\left(\mathbf{Z}_k | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right)$ is the marginal distribution of $P_\theta\left(\mathbf{Z}_k, \underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)} | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right)$ with respect to $\mathbf{Z}_k$, since $\underline{\mathbf{A}}_k^{(s)}$ and $\underline{\mathbf{A}}_k^{(t)}$ are integrated over some distribution. Herein we can impose another distribution of the topology $Q_\phi(\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)} | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)})$ characterized by parameter ϕ, then we have:

$$\begin{aligned}&\log \int_{\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)}} P_\theta\left(\mathbf{Z}_k, \underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)} | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right)\\ =&\log \int_{\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)}} P_\theta\left(\mathbf{Z}_k, \underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)} | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right) \frac{Q_\phi(\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)} | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)})}{Q_\phi(\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)} | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)})}\\ =&\log\left(\mathbb{E}_{Q_\phi(\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)} | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)})}\left[\frac{P_\theta\left(\mathbf{Z}_k, \underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)} | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right)}{Q_\phi(\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)} | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)})}\right]\right)\\ \geq&\mathbb{E}_{Q_\phi(\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)} | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)})}\Big[\log P_\theta(\mathbf{Z}, \underline{\mathbf{A}}^{(s)}, \underline{\mathbf{A}}^{(t)} | \mathcal{G}^{(s)}, \mathcal{G}^{(t)}) -\\ &\log Q_\phi(\underline{\mathbf{A}}^{(s)}, \underline{\mathbf{A}}^{(t)} | \mathcal{G}^{(s)}, \mathcal{G}^{(t)})\Big]\end{aligned} \tag{26}$$

where the final step is derived from Jensen's inequality. Since optimizating Eq. 25 is difficult, we can alter to maximize the right hand side of inequality of Eq. (26) instead, which is the Evidence Lower Bound (ELBO) (Bishop, 2006). Since two input graphs are handled separately by two identical subroutines (see Fig. 2(a)), we can then impose the independence of topology $\underline{\mathbf{A}}_k^{(s)}$ and $\underline{\mathbf{A}}_k^{(t)}$: $Q_\phi(\underline{\mathbf{A}}^{(s)}, \underline{\mathbf{A}}^{(t)} | \mathcal{G}^{(s)}, \mathcal{G}^{(t)}) = Q_\phi(\underline{\mathbf{A}}^{(s)} | \mathcal{G}^{(s)}) Q_\phi(\underline{\mathbf{A}}^{(t)} | \mathcal{G}^{(t)})$. In this sense, we can utilize the same parameter ϕ to characterize two identical neural networks (generators) for modeling Q_ϕ.

Assuming θ is fixed, ELBO is determined by Q_ϕ. According to Jensen's inequality, equality of Eq. (26) holds when:

$$\frac{P_\theta\left(\mathbf{Z}_k, \underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)} | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right)}{Q_\phi\left(\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)} | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right)} = c \tag{27}$$

where $c \neq 0$ is a constant. We then have:

$$\begin{aligned}&\int_{\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)}} P_\theta\left(\mathbf{Z}_k, \underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)} | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right)\\ =c&\int_{\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)}} Q_\phi\left(\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)} | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right)\end{aligned} \tag{28}$$

As Q_ϕ is a distribution, we have:

$$\int_{\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)}} Q_\phi\left(\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)} | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right) = 1 \tag{29}$$

Therefore, we have:

$$\int_{\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)}} P_\theta\left(\mathbf{Z}_k, \underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)} | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right) = c \tag{30}$$

We now have:

$$\begin{aligned}&Q_\phi\left(\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)} | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right)\\ =&\frac{P_\theta\left(\mathbf{Z}_k, \underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)} | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right)}{c}\\ =&\frac{P_\theta\left(\mathbf{Z}_k, \underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)} | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right)}{\int_{\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)}} P_\theta\left(\mathbf{Z}_k, \underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)} | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right)}\\ =&\frac{P_\theta\left(\mathbf{Z}_k, \underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)} | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right)}{P_\theta\left(\mathbf{Z}_k | \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right)}\\ =&P_\theta\left(\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)} | \mathbf{Z}_k, \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right)\end{aligned} \tag{31}$$

Eq. (31) shows that, once θ is fixed, maximizing ELBO amounts to finding a distribution Q_ϕ approximating the posterior probability $P_\theta\left(\underline{\mathbf{A}}_k^{(s)}, \underline{\mathbf{A}}_k^{(t)} | \mathbf{Z}_k, \mathcal{G}_k^{(s)}, \mathcal{G}_k^{(t)}\right)$. This can be done by training the generator Q_ϕ to produce latent topology $\underline{\mathbf{A}}$ given graph pair and the matching $\mathbf{Z}$. This corresponds to the **Inference** part in Sec. 3.4.2. Ablation study In this part, we evaluate the performance of DLGM-D and DLGM-G by selectively deactivating different loss functions (refer Sec. 3.3 for more details of the functions). We also conduct the test on DLGM-G using different sample size of the generator. This ablation test is conducted on

Pascal VOC dataset and average accuracy is reported in Tab. 5 and 6.

We first test the performance of both settings of DLGM by selectively activate the designated loss functions. Experimental results are summarized in Tab. 5. As matching loss $\mathcal{L}_M$ is essential for GM task, we constantly activate this loss for all settings. We see that the proposed novel losses $\mathcal{L}_C$ and $\mathcal{L}_L$ can consistently enhance the matching performance. Besides, DLGM-G indeed delivers better performance than DLGM-D under fair comparison.

We then test the impact of sample size from the generator Q_ϕ under DLGM-G. Experimental results are summarized in Tab. 6. We see that along with the increasing sample size, the average accuracy ascends. The performance becomes stable when the sample size reaches over 16.

A.5. More visual examples and analysis

We show more visual examples of matchings and generated topology using DLGM-G on Pascal VOC in Tab. 8 and Tab. 9, respectively. Each table follows **distinct coloring regulation** which will be detailed as follows:

- Tab. 8. For each class, the left and right images corresponds to Delaunay triangulation. The image in the middle refers to the predicted matching and generated graph topology. Cyan solid and dashed lines correspond to correct and wrong matchings, respectively. Green dashed lines are the ground-truth matchings that are missed by our model.

- Tab. 9. In this table, the leftmost and the rightmost columns correspond to original topology constructed using Delaunay triangulation. The two columns in the middle are the generated topology using our method given Delaunay triangulation as prior. Blue edges are the edges that Delaunay and generated ones have in common. Green edges corresponds to the ones that are in Delaunay but not in generated topology, while red edges are the ones that are generated but not in Delaunay.

We give some analysis for the following questions.

In what case a different graph is generated?

Since there are some generated graphs are identical to Delaunay, this question may naturally arise. We observe that, DLGM tends to produce an identical graph to Delaunay when objects are rarely with distortion and graphs are simple (e.g. tv, bottle and plant in Tab. 8 and last two rows in Tab. 9). However, when Delaunay is not sufficient to reveal the complex geometric relation or objects are with large distortion and feature diversity (e.g. cow and cat in Tab. 8 and person in Tab. 9), DLGM will resort to generating new topology with richer and stronger hint for graph matching. In other words, DLGM somewhat finds a way to identify if current instance pair is difficult or easy to match, and learns an adaptive strategy to handle these two cases.

Why DLGM-G delivers better performance than DLGM-D?

In general, DLGM-D is a deterministic gradient-based method. That is, the solution trajectory of DLGM-D almost follows the gradient direction at each iteration (with some variance from mini-batch). Though it is assured to reach a local optima, only following gradient is too greedy since generated graph is coupled with predicted matching. Besides, as the topology is discrete, the optimal continuous solution will have a large objective score gap to its nearest discrete sampled solution once the landspace of the neural network is too sharp. On the other hand, DLGM-G performs discrete sampling under feasible graph distribution at each iteration, which generally but not fully follows the gradient direction. This procedure can thus find better *discrete direction* with probability, hence better exploring the searching space. This behavior is similar to Reinforcement Learning, but with much higher efficiency. Additionally, EM framework can guarantee the convergence (Bishop, 2006).

Table 8. Matching examples of DLGM-G on 20 classes of Pascal VOC. The coloring of graphs and matchings follows the principle of Fig. 1 in the manuscript. Zoom in for better view.

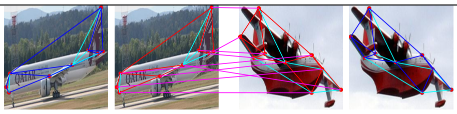
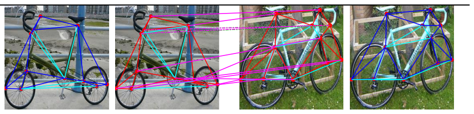
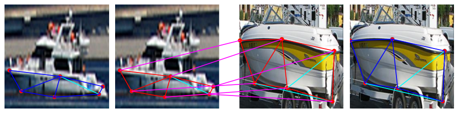
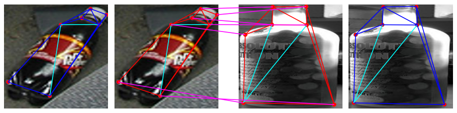
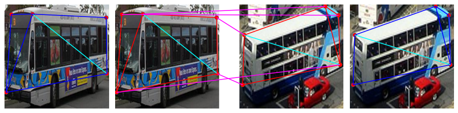
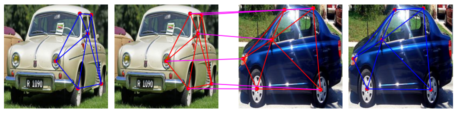
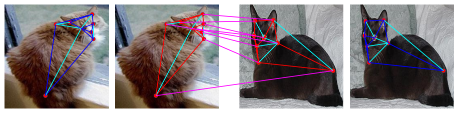
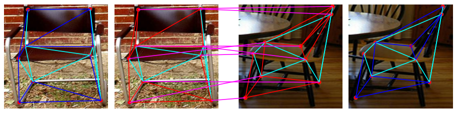
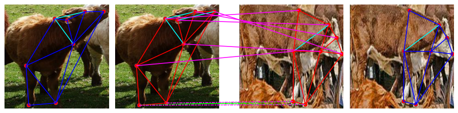
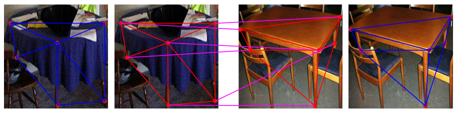
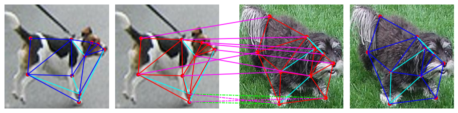
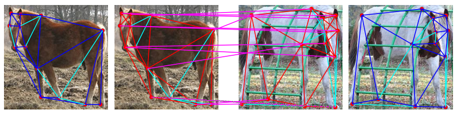
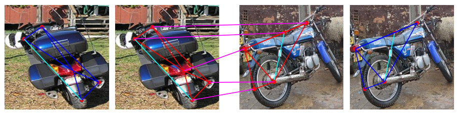
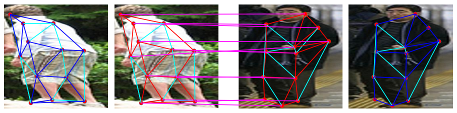
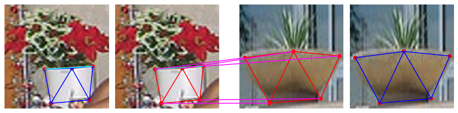
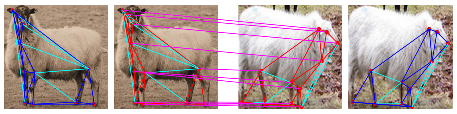
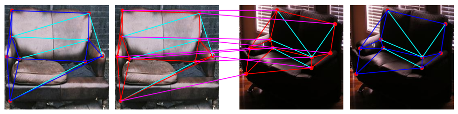
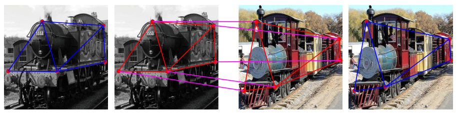
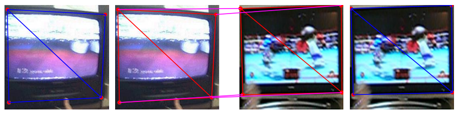
CLASS		
AERO & BIKE		
BIRD & BOAT		
BOTTLE & BUS		
CAR & CAT		
CHAIR & COW		
TABLE & DOG		
HORSE & MBIKE		
PERSON & PLANT		
SHEEP & SOFA		
TRAIN & TV		

Table 9. Generated topology compared with original Delaunay triangulation in a pairwise fashion. Note the 1st and the 4th columns correspond to two input images with topology constructed by Delaunay triangulation, respectively. 2nd and 3rd columns are the generated topology given Delaunay results as prior.

DELAUNAY 1	GENERATED 1	GENERATED 2	DELAUNAY 2