All Graphs Lead to Rome: Learning Geometric and Cycle-Consistent Representations with Graph Convolutional Networks

Stephen Phillips, Kostas Daniilidis
GRASP Laboratory, University of Pennsylvania
{stephi, kostas}@seas.upenn.edu

Abstract

Image feature matching is a fundamental part of many geometric computer vision applications, and using multiple images can improve performance. In this work, we formulate multi-image matching as a graph embedding problem then use a Graph Convolutional Network to learn an appropriate embedding function for aligning image features. We use cycle consistency to train our network in an unsupervised fashion, since ground truth correspondence is difficult or expensive to aquire. In addition, geometric consistency losses can be added at training time, even if the information is not available in the test set, unlike previous approaches that optimize cycle consistency directly. To the best of our knowledge, no other works have used learning for multi-image feature matching. Our experiments show that our method is competitive with other optimization based approaches.

1. Introduction

Feature matching is an essential part of Structure from Motion and many geometric computer vision applications. The goal in multi-image feature matching is to take 2D image locations from three or more images and find which ones correspond to the same point in the 3D scene. Methods such as SIFT feature matching [19] followed by RANSAC [11] have been the standard for decades. However RANSAC-based approaches are limited to matching pairs of images, which can lead to global inconsistencies in the matching. Other works, such as Wang et al. [26], have shown improvement in performance by optimizing cycle consistency, i.e. enforcing the pairwise feature matches to be globally consistent, illustrated in figure 1.

However, these multi-view consistency algorithms struggle in distributed and robust settings. Deep learning has revolutionized how image features are computed [28]. In this paper, we want to leverage the power of deep representations in order to compute feature descriptors that are robust
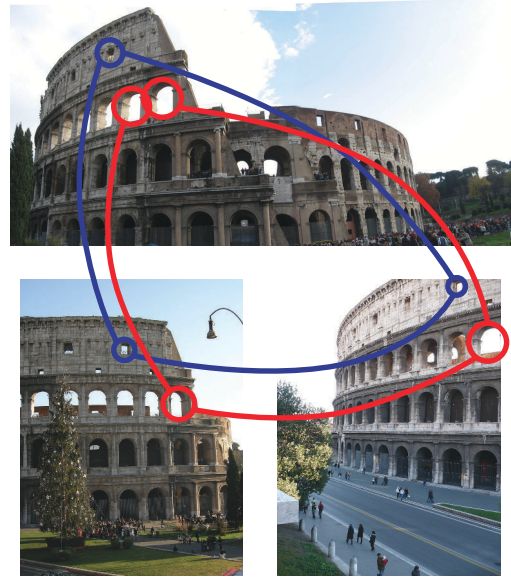

Figure 1. Example of multi-image matching and cycle consistency, with images from Rome16K dataset [18]. Pairwise matches between image can lead to global inconsistencies in the matching. In this example, an error in matching leads to an inconsistent cycle (shown in red), ideally these pairwise matches would be consistent. We improve the quality of pairwise matches in the multi-view matching problem by training a neural net with cycle consistency.

across multiple views.

Unfortunately, there are obstacles to applying multi-view constraints directly to deep learning. To train networks, we need a large amount of labeled data. In the case of multi-image feature matching, one would need hand labeled point correspondences between images, which are difficult and expensive to obtain. Multi-view constraints are formulated in terms of sparse features, which traditional convolutional neural nets are not designed to handle. Additionally, in the case of multi-image feature matching, geometric constraints would be helpful in rejecting outlier matches.

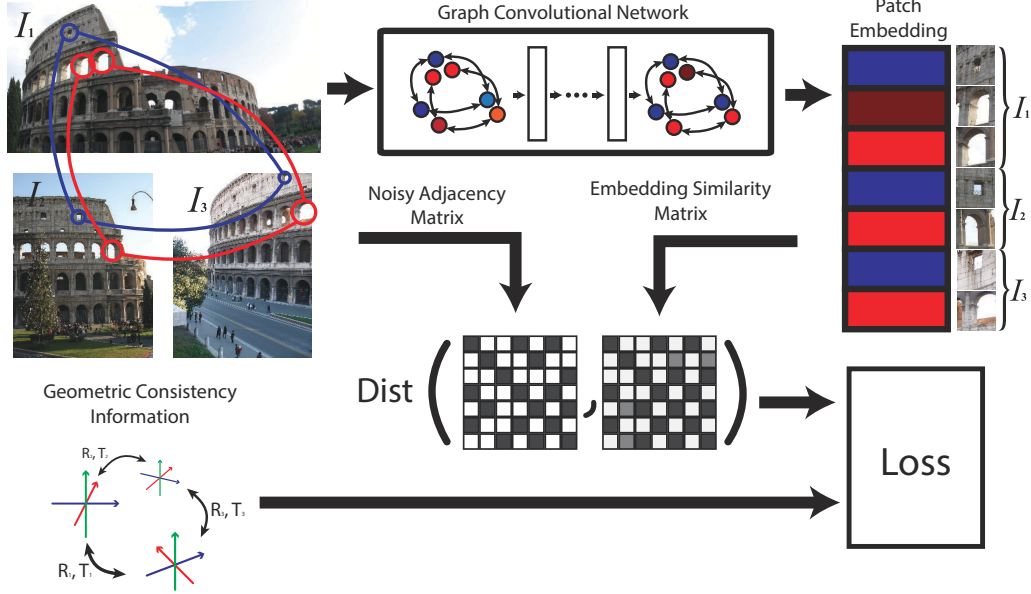In this work, we propose to solve these problems using a

Figure 2. An illustration of the approach of this work. The Graph Convolutional Neural Network (GCN) [16] takes as input the graph of matches and then outputs a low rank embedding of the adjacency matrix of the graph. The GCN operates on an embedding over the nodes of the graph. In the figure, the GCN node embeddings are represented by different colors. The final embedding is used to construct a pairwise similarity matrix, which we train to be a low dimensional cycle-consistent representation of the graph adjacency matrix, thus pruning the erroneous matches. We train the network using a reconstruction loss on the similarity matrix with the noisy adjacency matrix, and thus do not need ground truth matches. In addition, we can use geometric consistency information, such as epipolar constraints, to assist training the network.

Graph Convolutional Network (GCN) to operate on the correspondence graph. The proposed method works directly on the correspondence graph, which is agnostic to how the correspondences were computed, thus allowing the algorithm to work in a broad class of environments. To the best of our knowledge this work is the first to apply deep learning to the multi-view feature matching problem. We use an unsupervised loss - the cycle consistency loss - to train the network, and thus avoiding the difficulty of expensive hand labeling. Geometric consistency losses can aid training, even if such information is not available at test time. Although our network is simple, it shows promising results compared to baselines which optimize for cycle-consistency without learned embeddings. Furthermore, since inference requires only a single forward pass over the neural network, our approach is faster (to achieve comparable accuracy) than methods which must solve an optimization problem every time. Our contributions are:

- We use a novel architecture to address the multi-image feature matching problem using GCNs with graph embeddings.

- We introduce an unsupervised cycle consistency loss that does not require labeled correspondences to train.

- We demonstrate the effectiveness of geometric consis-

tency losses in improving training.

- We perform experiments on the Rome16K [18] dataset to test the effectiveness of our method compared to optimization based methods.

2. Related Work

2.1. Feature Matching

Matching has a rich history of research in computer vision. The most well known and widely used method for image matching is RANSAC [11]. RANSAC, combined with hand-crafted feature descriptors such as SIFT [19], SURF [3], BRIEF [8], or ORB [20], has constituted the bulk of the matching literature for the last 40 years. Finally, graph matching [23, 14] can be used as a final step for more robust matches.

2.2. Multi-image Matching

Multi-image matching has traditionally been done using optimization based methods minimizing a cycle consistency based loss (see Section 3.3). Pachauri et al. [21] and Arrigoni et al. [1] use the eigenvectors of the matching matrix to obtain a low dimensional embedding. However, the low Gaussian noise assumption is not realistic. Zhou et al. [31] and Wang et al. [26] use most sophisticated optimization

techniques on the matching matrix and thus produce more robust solutions. Leonardos et al. [17] implement a distributed optimization scheme to solve for cycle consistency. As an alternative to optimization based techniques, Tron et al. [25] used density based clustering techniques to compute multi-image correspondence. To the best of our knowledge, we are the first to use neural networks for multi-image matching.

2.3. Deep Learning for Matching

Previous attempts to improve image matching techniques using machine learning have focused on learning the descriptors given ground truth correspondence from curated datasets [30, 28, 4]. This approach is limited if one do not have the ability to get the ground truth correspondences. There are other methods to build correspondences such as Choy et al. [9], but they only handle two-view constraints and require dense correspondences. Most similar to our work, Yi et al. [29] attempts to improve correspondences by learning match probabilities for RANSAC for greater robustness and speed. However, they only focus on two view matching and do not exploit the advantages of the correspondence structure. Note that while Zhu et al. [32] use cycle consistency in their loss, their method is restricted to pairwise cycle consistency, while we use multi-image cycle consistency.

2.4. Graph Neural Networks

Graph neural networks have recieved more attention recently [6, 7, 10, 16, 22, 13, 12, 2]. Classical so-called Spectral methods used the eigenvectors of the graph Laplacian to compute convolutions as in Bruna et al. [7], but requires an a-priori known graph structure. Non-spectral methods do not require a-priori knowledge [6, 16, 22, 12]. Most of these methods use polynomials of the graph Laplacian to compute neighborhood averages. Gama et al. 2018 [13, 12] formalize this notion and generalize it beyond the use of the graph Laplacian. To improve performance, more sophisticated aggregation techniques and global information passing can be used as discussed in Battaglia et al. [2].

3. Method

Our goal is to learn optimal features that capture multiple image views by filtering out noisy feature matches. The input to our algorithm is a set of features and noisy correspondences, and the output is a new set of features where the pairwise similarities of these features correspond to the true matches. We do this by training the new set of feature embeddings to be cycle consistent. We formulate this problem in terms of the correspondence graph of the features. Vectors $\mathbf{x}$ and matrices $\mathbf{A}$ are denoted with boldface, and the ith row of a matrix is denoted $(\mathbf{A})_i$.
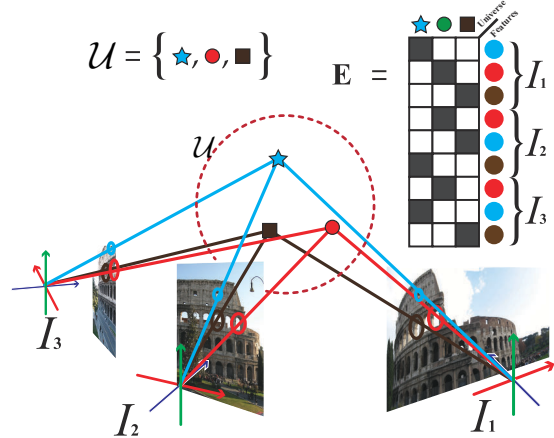

Figure 3. Illustration of the universe of features. Each feature in each image corresponds to a 3D point in the scene. We can construct cycle consistent embeddings of the features by mapping each one to the one-hot vector of its corresponding 3D point. While there can be many features, there are fewer 3D points and thus this corresponds to a low rank factorization of the correspondence matrix. Best viewed in color.

3.1. Correspondence Graph

We assume there is an initial set of feature matches represented as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Each vertex v of the graph is a feature with its associated descriptor f_v. The graph is represented by its Weighted Adjacency Matrix $\mathbf{A}(\mathcal{G}) \in \mathbb{R}^{n \times n}$, where $(\mathbf{A}(\mathcal{G}))_{ij}$ is the strength of the match between nodes i and j. We introduce the positive diagonal degree matrix $\mathbf{D}(\mathcal{G}) \in \mathbb{R}^{n \times n}$, with $\mathbf{D}(\mathcal{G})_{ii} = \sum_j (\mathbf{A}(\mathcal{G}))_{ij}$. For brevity we denote these matrices $\mathbf{A}$ and $\mathbf{D}$ respectively. We introduce the notion of an embedding matrix $\mathbf{E}_0$ by concatenating the features f_v from each of the vertices. We use $\mathbf{E}_0$ as the initialization for our learning algorithm. We add any additional knowledge we have into the embedding as well (e.g. scale, orientation, etc.). Putative correspondences are computed from these features and represented by weighted edges in which the weight gives the strength of the match. While there are many interesting methods for computing these putative correspondences [23, 29], we do not explore them in this work.

In the absence of noise or outliers, this graph would have a connected component for each visible point in the world. These components should be mutually disjoint. In other words each feature would only have edges to other features corresponding to the same point. Since features in this case represent unique locations in the scene, no points in the same image would have edges between them. In the noisy case we expect this structure to be corrupted and thus we need to prune the erroneous edges.

To compute this matching graph, we do pairwise feature

matching between the images, creating putative correspondences for each of the features. Typically these putative correspondences are matched probabilistically, meaning a feature in one image matches to many features in another. The ambiguity in the matches could come from repeated structures in the scene, insufficiently informative low-level feature descriptors, or just an error in the matching algorithm. Filtering out these noisy matches is our primary learning goal.

However, this graph structure does not have a regular grid structure and thus we cannot use standard convolutional nets to learn features for this task. Instead we use graph convolutions to learn feature representations on this space. We describe our approach in the next section.

3.2. Graph Neural Networks

As input to our method we are given the graph Laplacian $\mathbf{L} \in \mathbb{R}^{n \times n}$ to encode the graph structure:

$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} \tag{1}$$

and an initial embedding $\mathbf{E}_0 \in \mathbb{R}^{n \times m_0}$. Many older methods assume $\mathbf{L}$ is known a-priori [7], and can encode graph convolutions using the eigenvectors of L. We do not have this luxury, as the correspondence structure changes from image to image, and thus we use non-spectral Graph Neural Networks. The standard form of non-spectral Graph Neural Networks is:

$$\mathbf{E}_{i+1} = P\left(\sigma\left(\sum_k \mathbf{L}^k \mathbf{E}_i \mathbf{W}_{i,k}\right)\right) \tag{2}$$

Here σ is the non-linearity (typically a ReLU), $\mathbf{W}_{k,i}$ are the learned weights of layer i, and P is the 'pooling' operation, known as graph coarsening [6, 13]. The maximum degree of $\mathbf{L}$ determines the number of hops away from a node one layer can access. In the feature matching problem, we require an embedding for each vertex. Therefore we cannot perform graph coarsening, and thus only perform the non-linearity step. This work uses Graph Convolutional Networks, proposed in Kipf and Welling [16]:

$$\widetilde{\mathbf{L}} = (\mathbf{D} + \mathbf{I})^{-1/2}(\mathbf{A} + \mathbf{I})(\mathbf{D} + \mathbf{I})^{-1/2} \tag{3}$$

$$\mathbf{E}_{l+1} = \sigma\left(\widetilde{\mathbf{L}}\mathbf{E}_i \mathbf{W}_i\right) \tag{4}$$

The matrix $\widetilde{\mathbf{L}}$ is analogous to the graph Laplacian (with better numerical stability properties). $\widetilde{\mathbf{L}}$ encodes the structure of the graph and is used to perform the actual graph convolution. Our network has many layers, and thus the receptive field of the final embedding is quite large, even without the pooling operations. In principle we could use higher hop neighborhoods [12], or more complicated aggregation structures [2], but in this work we restrict ourselves
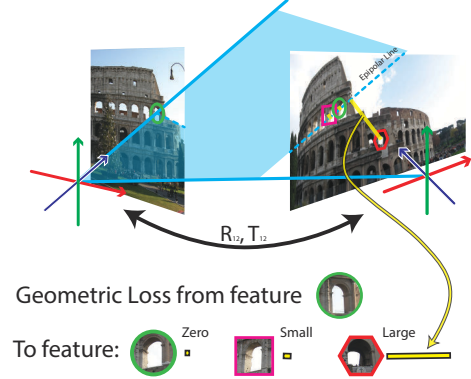

Figure 4. Illustrated here is an example of how the geometric loss is computed for one feature. Errors are computed via absolute distance from the epipolar line, as expressed by (10) via the epipolar constraint. The epipolar line is the line of projection of the feature on the first image, projected onto to the second. The distance to this line on the second image indicates how likely that point is to correspond geometrically to the original feature. There can be false positives along the projected line, as shown by the square feature in the figure, but other points will be eliminated, such as the hexagonal feature. Best viewed in color.

to simpler architectures, and still obtain promising results for this task. For faster training, we use group normalization [27] and skip connections. The final output $\mathbf{E}_n$ gives us a new descriptor for every node. In the next sections we will discuss the loss functions to train this network.

3.3. Cycle Consistency

Let $\mathbf{M}$ be the noiseless set of matches between our features, with $\mathbf{M}_{ij}$ being the matches between image i and image j. If the pairwise matches are globally consistent, then we can say that, for all i, j, k:

$$\mathbf{M}_{ij} = \mathbf{M}_{ik}\mathbf{M}_{kj} \tag{5}$$

In other words, the matches between two images stay the same no matter what path is taken to get there. This constraint is known as *cycle consistency*, and has been used in a number of works to optimize for global consistency [31, 26, 17]. Stated in this form, there are $O(n^3)$ cycle consistency constraints to check. A more elegant way to represent cycle consistency is to first create a 'universe' of features that all images match to (see figure 3). Then, one can match the ith set of features to the universe using a matrix $\mathbf{X}_i$. Then the cycle consistency becomes:

$$\mathbf{M}_{ij} = \mathbf{X}_i\mathbf{X}_j^\top \tag{6}$$

This reduces our complexity from $O(n^3)$ to $O(n^2)$. We try to learn $\mathbf{E}$ to approximate $\mathbf{X}$ - in other words the final

Method	Same Point Similarities	Different Point Similarities
Ideal	1.00e+0 $\pm$ 0.00e+0	0.00e+0 $\pm$ 0.00e+0
Initialization Baseline	5.11e-1 $\pm$ 1.68e-2	2.56e-1 $\pm$ 2.06e-1
3 Views, Noiseless	9.96e-1 $\pm$ 7.70e-3	1.16e-1 $\pm$ 1.32e-1
5 Views, Noiseless	1.00e+0 $\pm$ 4.15e-4	1.22e-1 $\pm$ 1.67e-1
3 Views, Added Noise	9.96e-1 $\pm$ 7.70e-3	1.16e-1 $\pm$ 1.32e-1
5 Views, Added Noise	9.89e-1 $\pm$ 2.47e-2	7.67e-2 $\pm$ 1.56e-1
6 Views, Added Noise	9.84e-1 $\pm$ 3.16e-2	7.46e-2 $\pm$ 1.57e-1
3 Views, 5% Outliers	9.29e-1 $\pm$ 1.79e-1	1.41e-1 $\pm$ 1.48e-1
3 Views, 10% Outliers	9.27e-1 $\pm$ 1.79e-1	1.40e-1 $\pm$ 1.51e-1

Table 1. Results on Synthetic correspondence graphs. The 'Same Point Similarities' column is the mean and standard deviation of similarities for true corresponding points, while the 'Different Point Similarities' is the same for points that do not correspond. For the 'Same Point Similarities' column higher is better, and for 'Different Point Similarities' lower is better. Losses tested against ground truth correspondence graph adjacency matrices. Our method was not trained on ground truth correspondences but using unsupervised methods.

embedding should be an encoding of the universe of features. As we do not have the ground truth matches $\mathbf{M}$, we approximate it using the noisy adjacency matrix $\mathbf{A}$ of our correspondence graph. Thus our loss would be

$$\mathcal{L}(\mathbf{A}, \mathbf{E}_n) = \mathcal{D}(\mathbf{A}, \mathbf{E}_n \mathbf{E}_n^\top) \qquad (7)$$

Here $\mathcal{D}$ could be an L_2 loss, L_1 loss, or many others. In this work, we use the L_1 loss. Note that because of this formulation, we can determine our embeddings only up to a rotation, as

$$\mathbf{E}R(\mathbf{E}R)^\top = \mathbf{E}RR^\top \mathbf{E}^\top = \mathbf{E}\mathbf{E}^\top \qquad (8)$$

Thus when visualizing embeddings, we rotate them to make them more interpretable (see figure 7).

3.4. Geometric Consistency Loss

One of the main advantages of this approach over more traditional optimization based approaches is the ability to add geometric consistency information into the loss at training time, even if it is not available at test time. The simplest way to add geometric consistency losses, and the approach we use here, is to use the epipolar constraint. The epipolar constraint describes how the positions of features in different images corresponding to the same point should be related. Given a relative pose (R_{ij}, T_{ij}) between two cameras i and j (transforms j to i) the epipolar on corresponding feature locations X_i and X_j:

$$X_i^\top [T_{ij}]_\times R_{ij} X_j = 0 \qquad (9)$$

In this work we use the two pose epipolar constraint [24]:

$$X_i^\top R_i^\top [T_j - T_i]_\times R_j X_j = 0 \qquad (10)$$

The constraint assumes that the X_k are calibrated i.e. the camera intrinsics are known. Given the matrix of ground

truth correspondences $\mathbf{M}_{ij}$ between camera i and j, we can formulate this as a loss:

$$\mathcal{L}_{ij,geom}(\mathbf{M}_{ij}) = \sum_{k,l} (\mathbf{M}_{ij})_{kl} \left| X_{i,k}^\top R_i^\top [T_j - T_i]_\times R_j X_{j,l} \right| \qquad (11)$$

For our purposes, since we use low rank embeddings $\mathbf{E}_i$, $\mathbf{E}_j$, the loss would read:

$$\mathcal{L}_{geom}(\mathbf{E}) = \text{tr}(\mathbf{G}^\top \mathbf{E}\mathbf{E}^\top) = \sum_{i,j} (\mathbf{E})_i \cdot (\mathbf{E})_j (\mathbf{G})_{ij} \qquad (12)$$

$$(\mathbf{G})_{ij} = \left| X_i^\top R_{c(i)}^\top [T_{c(j)} - T_{c(i)}]_\times R_{c(j)} X_j \right|$$

Where $c(i)$ is the appropriate camera for point index i.

4. Experiments

4.1. Synthetic Graph Dataset

We first test our method on synthetically generated data as a simple proof of concept. To generate the data, we generate p points, each with its own randomly generated descriptor. To create the graph, we generate random permutation matrices, with a noise applied to it after it is generated. We initialize the input descriptors using the true descriptor, plus some added Gaussian noise. No geometric losses were added during training for these experiments. However, the method was robust in testing with different noise functions and parameters. The normalized noisy input descriptors are our baseline - they correlate with the true values but do not preserve the structure well. However, the GCN recovered the true structure very well, as shown in Table 1. All experiments were run with a 12 layer GCN with the ReLU nonlinearity and skip connections. All were trained with the Adam optimizer [15] and a learning rate 10^{-4}

4.2. Rome 16K Graph Dataset

We use the Rome16K dataset [18] to test our algorithm in real world settings. Rome16K consists of 16 thousand

Method (3 Views)	L_1 Loss	L_2 Loss	Run Time (sec)
MatchALS [31] 15 Iterations	0.052 ± 0.003	0.010 ± 0.002	0.021 ± 0.003
MatchALS [31] 25 Iterations	0.045 ± 0.007	0.009 ± 0.003	0.034 ± 0.003
MatchALS [31] 50 Iterations	0.016 ± 0.008	0.007 ± 0.003	0.065 ± 0.006
PGDDS [17] 15 Iterations	0.016 ± 0.002	0.006 ± 0.002	0.287 ± 0.043
PGDDS [17] 25 Iterations	0.014 ± 0.002	0.005 ± 0.002	0.613 ± 0.089
PGDDS [17] 50 Iterations	0.013 ± 0.002	0.005 ± 0.002	1.430 ± 0.234
Spectral	0.054 ± 0.005	0.018 ± 0.004	0.018 ± 0.004
GCN, 12 Layers (ours)	**0.025 ± 0.003**	**0.016 ± 0.003**	**0.039 ± 0.009**
Method (4 Views)	L_1 Loss	L_2 Loss	Run Time (sec)
MatchALS [31] 15 Iterations	0.064 ± 0.005	0.012 ± 0.002	0.030 ± 0.004
MatchALS [31] 25 Iterations	0.041 ± 0.010	0.008 ± 0.004	0.048 ± 0.005
MatchALS [31] 50 Iterations	0.011 ± 0.008	0.005 ± 0.003	0.094 ± 0.008
PGDDS [17] 15 Iterations	0.015 ± 0.002	0.006 ± 0.001	0.436 ± 0.090
PGDDS [17] 25 Iterations	0.014 ± 0.002	0.005 ± 0.001	0.961 ± 0.181
PGDDS [17] 50 Iterations	0.013 ± 0.002	0.005 ± 0.002	2.056 ± 0.424
Spectral Method	0.055 ± 0.004	0.017 ± 0.003	0.028 ± 0.003
GCN, 12 Layers (ours)	**0.023 ± 0.003**	**0.015 ± 0.002**	**0.056 ± 0.017**

Table 2. Results on Rome16K Correspondence graphs, showing the mean and standard deviation of the L_1 and L_2. Our method was not trained on ground truth correspondences but using unsupervised methods and geometric side losses. As our method gives soft labels, we use cannot use precision or recall as is standard in testing cycle consistency [31]. Thus we test against ground truth correspondence graph adjacency matrices computed from the bundle adjustment output. Our method performs better than 25 iteration of the MatchALS [31] method, but does not perform as well as 50 iterations. We do not perform as well as the Projected Gradient Descent Doubly Stochastic (PGDDS) [17] but we perform significantly faster than them. Note that we perform much better in L_1 performance rather than L_2, as we optimized the network weights using an L_1 loss.
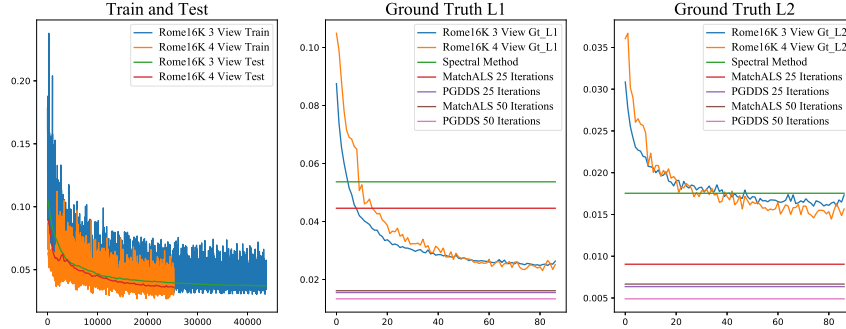

Figure 5. Training curves with and without Geometric Training loss. The geometric training loss improves testing performance. Shown as horizontal lines are the state of the art optimization based baselines. Even with a simple network we compare well with them.

images of various historical sites in Rome extracted from Flickr, along with the 3D structure of the sites provided by bundle adjustment. While not a standard dataset to test cycle consistency, other datasets had insufficient data to train a network on. Rome16K is typically used to test bundle adjustment methods. Therefore, to use our method, we extract image triplets and quadruplets with overlap of 80 points or more to test our algorithm, with the points established as corresponding in the given bundle adjustment output. For the initial embedding we use the original 128 dimensional

SIFT descriptors, normalized to have unit L_2 norm, the calibrated x-y position, the orientation, and log scale of the SIFT feature.

For these experiments we train with the L_1 norm and geometric consistency losses. We evaluate on a test set using the ground truth adjacency matrix, which we compute from the bundle adjustment given by the Rome16K dataset. Traditional methods of evaluation use Precision and Recall, after making the labels 0 or 1. We forgo this as our method was trained to output soft labels - we in turn make the labels
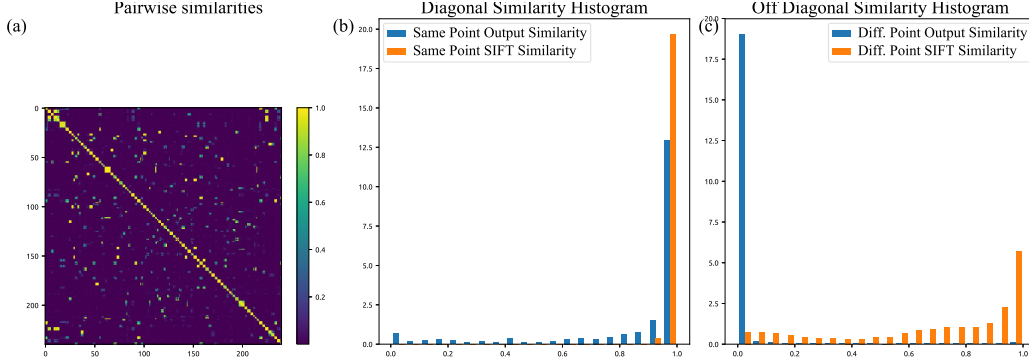
Figure 6. Example output of our network. (a) Similarity Matrix of the of the embeddings (b) Histogram of feature similarities for pairs which correspond to the same 3D point, comparing our output features with the original SIFT features (c) Histogram of feature similarities for pairs which correspond to different 3D points, comparing our output features with the original SIFT features

soft for the optimization based methods. For this method to work, we need the dimension of the embedding to be at least the number of unique points in the scene. Picking the correct number is difficult a-priori, and is a problem with all cycle consistency based methods. Here we use the ground truth dimension of the embedding to test both our method and the baselines. For implementation details, our network has 12 layers, and was trained with the Adam optimizer [15] with a learning rate of 10^{-4}, with an exponentially decaying learning rate. We incorporate skip connections between the input, 6th, and 12th layers, which greatly improved training convergence and testing performance (see supplemental material).

We compare our method to spectral and optimization based baselines with different maximum iteration cutoffs. Our method is signification better than a simple spectral based optimization. Our network, though only 12 layers, has comparable accuracy to MatchALS [31] run between 25 and 50 iterations. Although our method does not outperform the Projected Gradient Descent - Doubly Stochastic (PGDDS) [17] method, we have significantly better run time. Additional comparisons to different iterations numbers of iterations can be found in the supplementary material.

5. Conclusion

We have shown a novel method for training feature matching using GCNs, using an unsupervised cycle consistency loss and geometric consistency losses. We have demonstrated its effectiveness on the Rome16K dataset compared to the traditional optimization based baselines on a simple GCN. For future work, we will investigate robust losses for better outlier rejection, and using higher order geometric constraints, such as the tri-focal tensor, as additional loss terms. A long term goal is to incorporate learning image level features into this pipeline. Moreover we hope
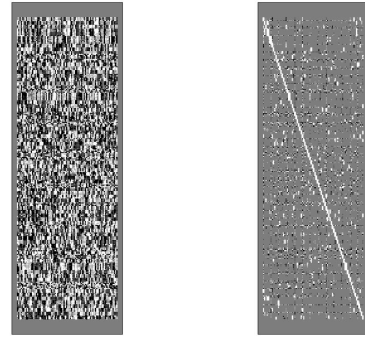

Figure 7. Visualization of the learned embeddings using Hinton diagrams [5]. On the left we have the raw outputs, which are difficult to interpret. On the right, we rotated the features to best match the ground truth for a more interpretable visualization.

to adapt this method to a distributed setting, which would be a simple extention given current approach.

References

[1] F. Arrigoni, E. Maset, and A. Fusiello. Synchronization in the symmetric inverse semigroup. In *International Conference on Image Analysis and Processing*, pages 70–81. Springer, 2017.

[2] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, et al. Relational inductive

biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[3] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[4] E. Brachmann, A. Krull, S. Nowozin, J. Shotton, F. Michel, S. Gumhold, and C. Rother. Dsac-differentiable ransac for camera localization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 3, 2017.

[5] F. J. Bremner, S. J. Gotts, and D. L. Denham. Hinton diagrams: Viewing connection strengths in neural networks. *Behavior Research Methods, Instruments, & Computers*, 26(2):215–218, 1994.

[6] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[7] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[8] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua. Brief: Computing a local binary descriptor very fast. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1281–1298, 2012.

[9] C. B. Choy, J. Gwak, S. Savarese, and M. Chandraker. Universal correspondence network. In *Advances in Neural Information Processing Systems*, pages 2414–2422, 2016.

[10] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852, 2016.

[11] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[12] F. Gama, G. Leus, A. G. Marques, and A. Ribeiro. Convolutional neural networks via node-varying graph filters. In *2018 IEEE Data Science Workshop (DSW)*, pages 1–5. IEEE, 2018.

[13] F. Gama, A. G. Marques, A. Ribeiro, and G. Leus. Mimo graph filters for convolutional neural networks. *arXiv preprint arXiv:1803.02247*, 2018.

[14] N. Hu, B. Thibert, and L. Guibas. Distributable consistent multi-graph matching. *arXiv preprint arXiv:1611.07191*, 2016.

[15] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[16] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[17] S. Leonardos, X. Zhou, and K. Daniilidis. Distributed consistent data association. *arXiv preprint arXiv:1609.07015*, 2016.

[18] Y. Li, N. Snavely, and D. P. Huttenlocher. Location recognition using prioritized feature matching. In *European conference on computer vision*, pages 791–804. Springer, 2010.

[19] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[20] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[21] D. Pachauri, R. Kondor, and V. Singh. Solving the multi-way matching problem by permutation synchronization. In *Advances in neural information processing systems*, pages 1860–1868, 2013.

[22] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[23] Y. Suh, K. Adamczewski, and K. Mu Lee. Subgraph matching using compactness prior for robust feature correspondence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5070–5078, 2015.

[24] R. Tron and K. Daniilidis. On the quotient representation for the essential manifold. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1574–1581, 2014.

[25] R. Tron, X. Zhou, C. Esteves, and K. Daniilidis. Fast multi-image matching via density-based clustering. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 4057–4066, 2017.

[26] Q. Wang, X. Zhou, and K. Daniilidis. Multi-image semantic matching by mining consistent features. *arXiv preprint arXiv:1711.07641*, 2017.

[27] Y. Wu and K. He. Group normalization. *arXiv preprint arXiv:1803.08494*, 2018.

[28] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua. Lift: Learned invariant feature transform. In *European Conference on Computer Vision*, pages 467–483. Springer, 2016.

[29] K. M. Yi, E. Trulls, Y. Ono, V. Lepetit, M. Salzmann, and P. Fua. Learning to find good correspondences. In *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, number CONF, 2018.

[30] S. Zagoruyko and N. Komodakis. Learning to compare image patches via convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4353–4361, 2015.

[31] X. Zhou, M. Zhu, and K. Daniilidis. Multi-image matching via fast alternating minimization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4032–4040, 2015.

[32] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint*, 2017.

Appendix

A. Relative Efficiency Against Optimization Based Methods

In figure 8 we show a more fine grained comparison of our algorithm to optimization based methods at different numbers of iterations. We show the mean and standard devation of the errors of all methods, testing the optimization based methods at different iterations.
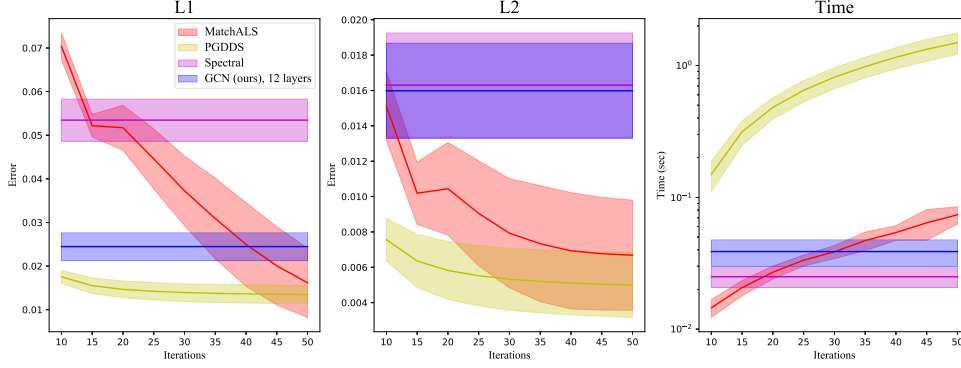

Figure 8. This figure plots the average accuracy of the optimization methods by iteration compared to 12 iterations of our GCN based method. The bold center line shows the mean, the colored in region shows 1 standard devation of the errors. As we optimize for L_1 error, our method's L_1 loss is substantially better than its L_2 losses. The right most plot shows time, plotted on a log scale for clarity.

B. Effect of Group Normalization

We observe that training with group normalization helps with training and generalization error, as shown in figure 9. As the Group Normalization operation is not easily parallelized across nodes in the graph, the use of Group Normalization is the main barrier to parallelization. Thus we will need to find alternative more parallelizable methods to improve generalization error for future work.
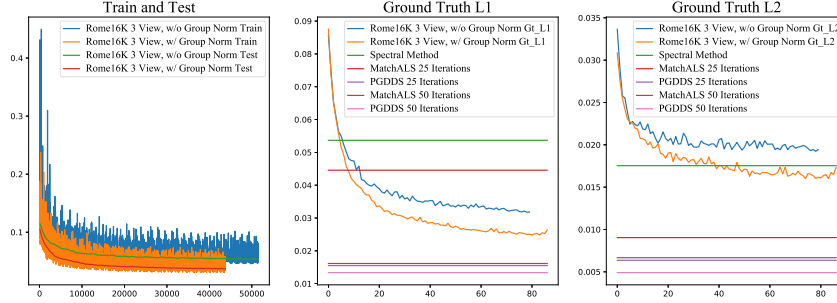

Figure 9. This figure plots the training curves of the network with and without Group Normalization. Note that while the train/test curves are fairly similar with or without Group Normalization, distance to the ground truth is improved greatly with Group Normalization.