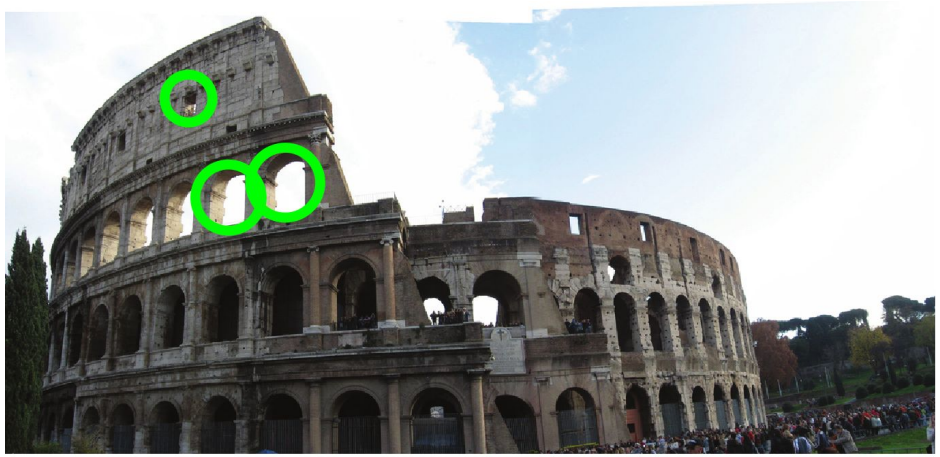
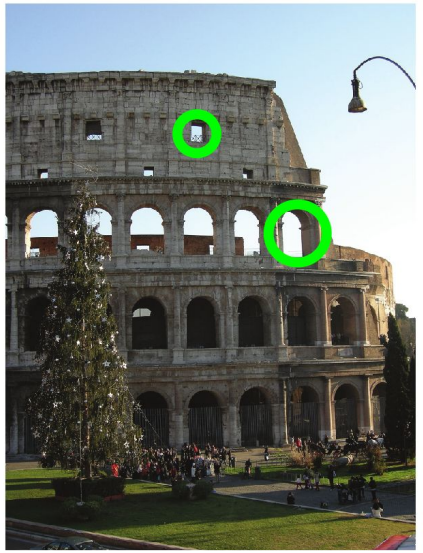
All Graphs Lead to Rome: Learning Geometric and Cycle-Consistent Representations with Graph Convolutional Networks

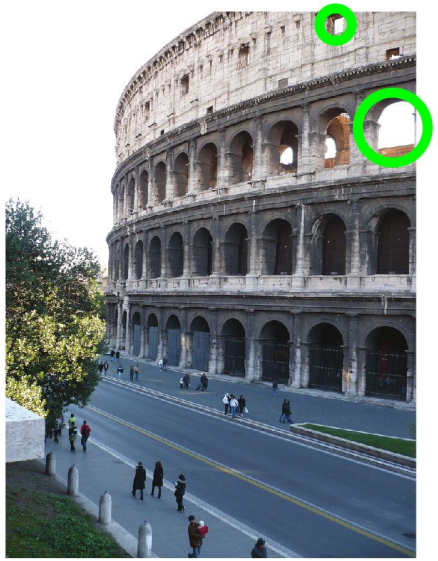
Stephen Phillips
Kostas Daniilidis

Motivation

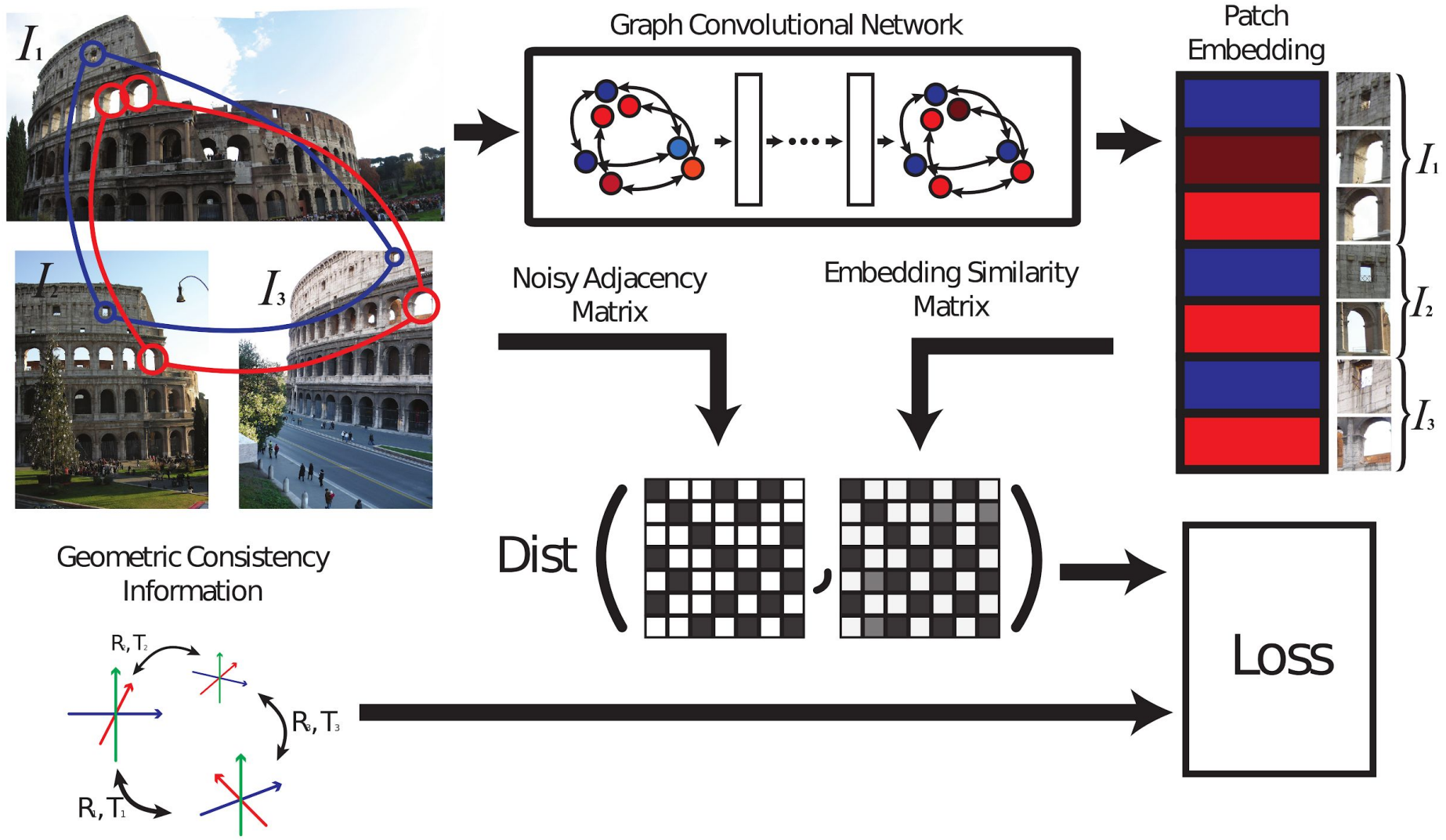
- Finding correspondence between radically different viewpoints is difficult
- Hand-crafted local appearance models don’t work across such large variations
- Machine Learning shows promise in creating representations that can handle such variations
- Challenges:
  - Sparse correspondences in DNNs, Lack of labelled data, End to end learning





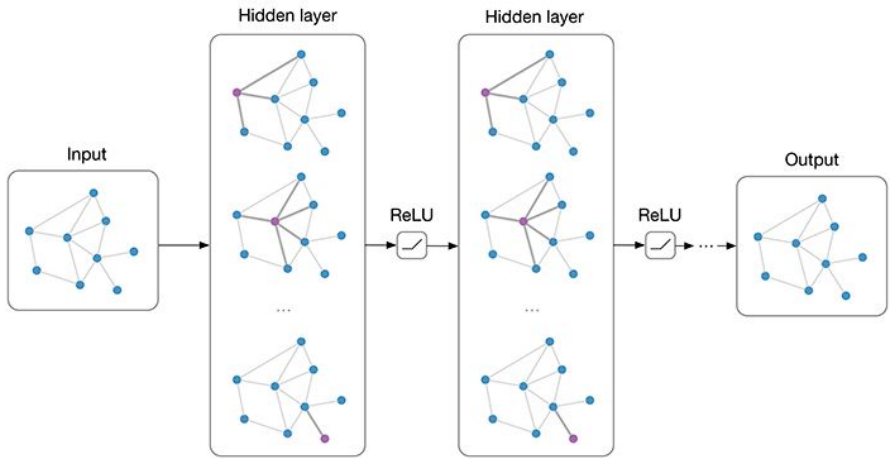


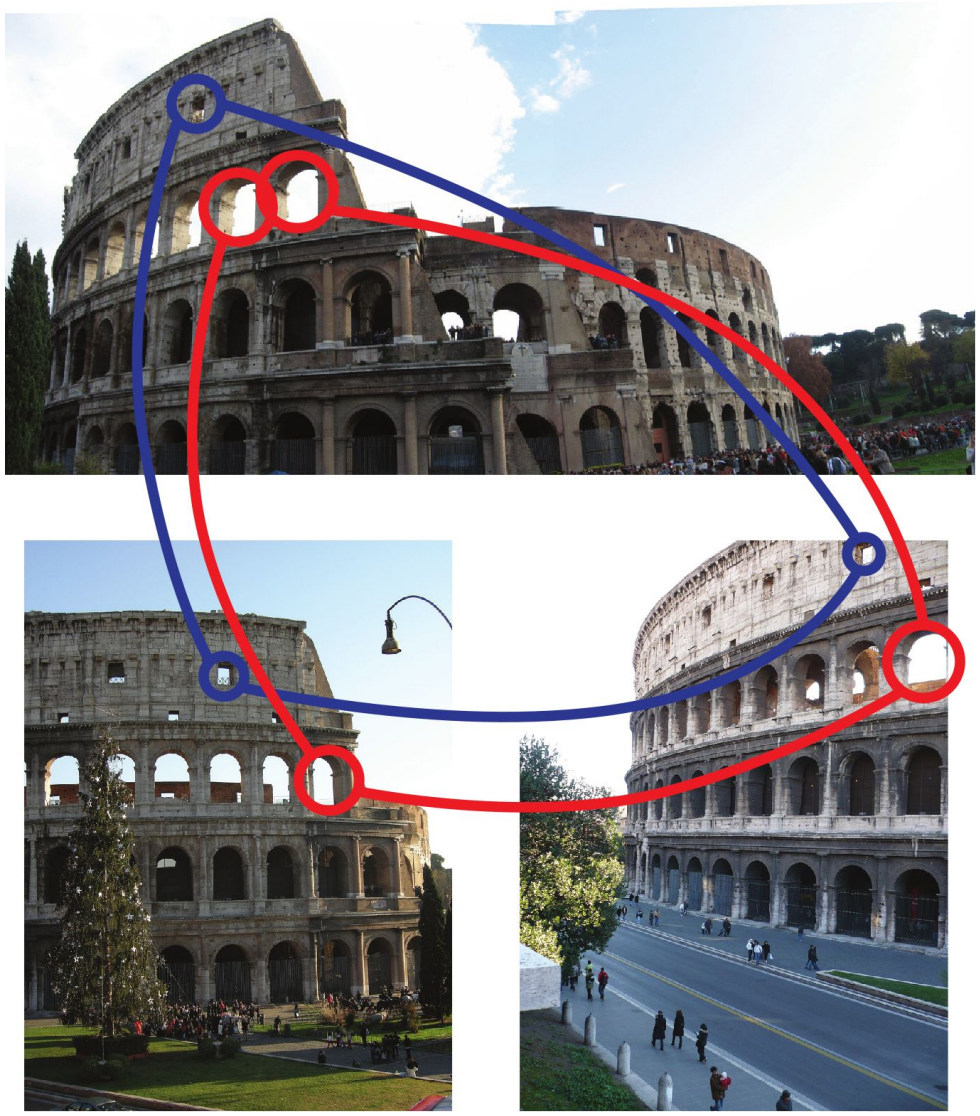
Outline


I_1
I_2
I_3
Graph Convolutional Network
Patch Embedding
I_1
I_2
I_3
Noisy Adjacency Matrix
Embedding Similarity Matrix
Dist
Loss
Geometric Consistency Information
R, T
R, T
R, T

Correspondences as a Graph

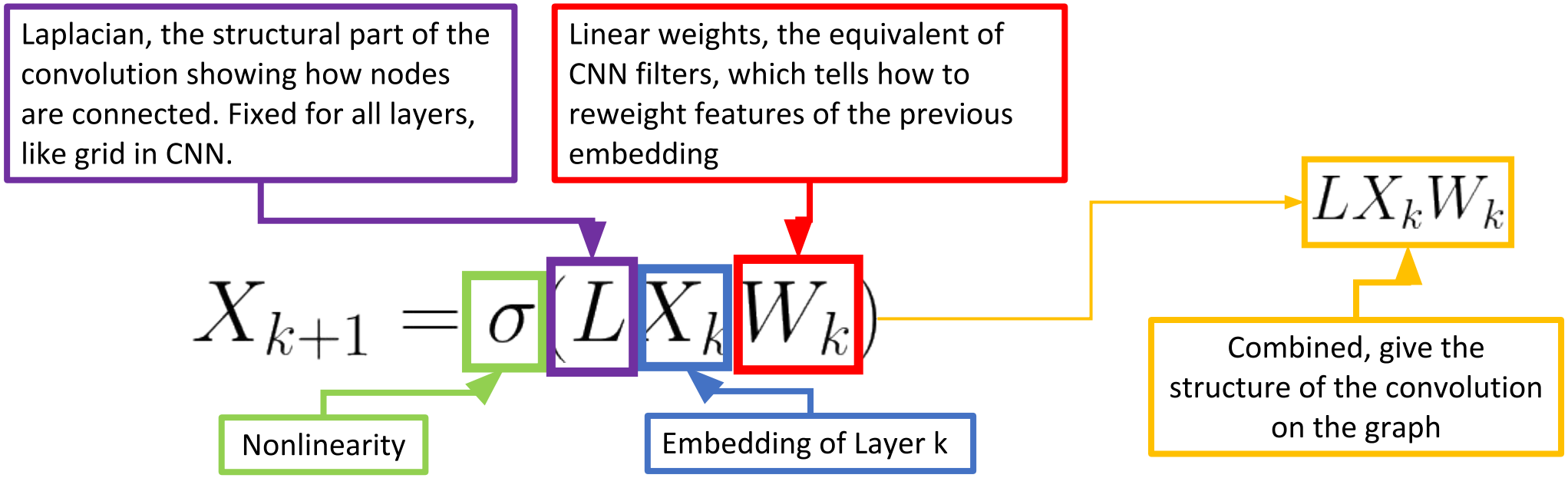
- Can treat a correspondences as a graph
  - Each Node represents a point on the image, with possibly a feature attached to it
  - Each edge represents a putative correspondence
  - Potential errors in the correspondence
  - Can use Graph Convolutional Nets





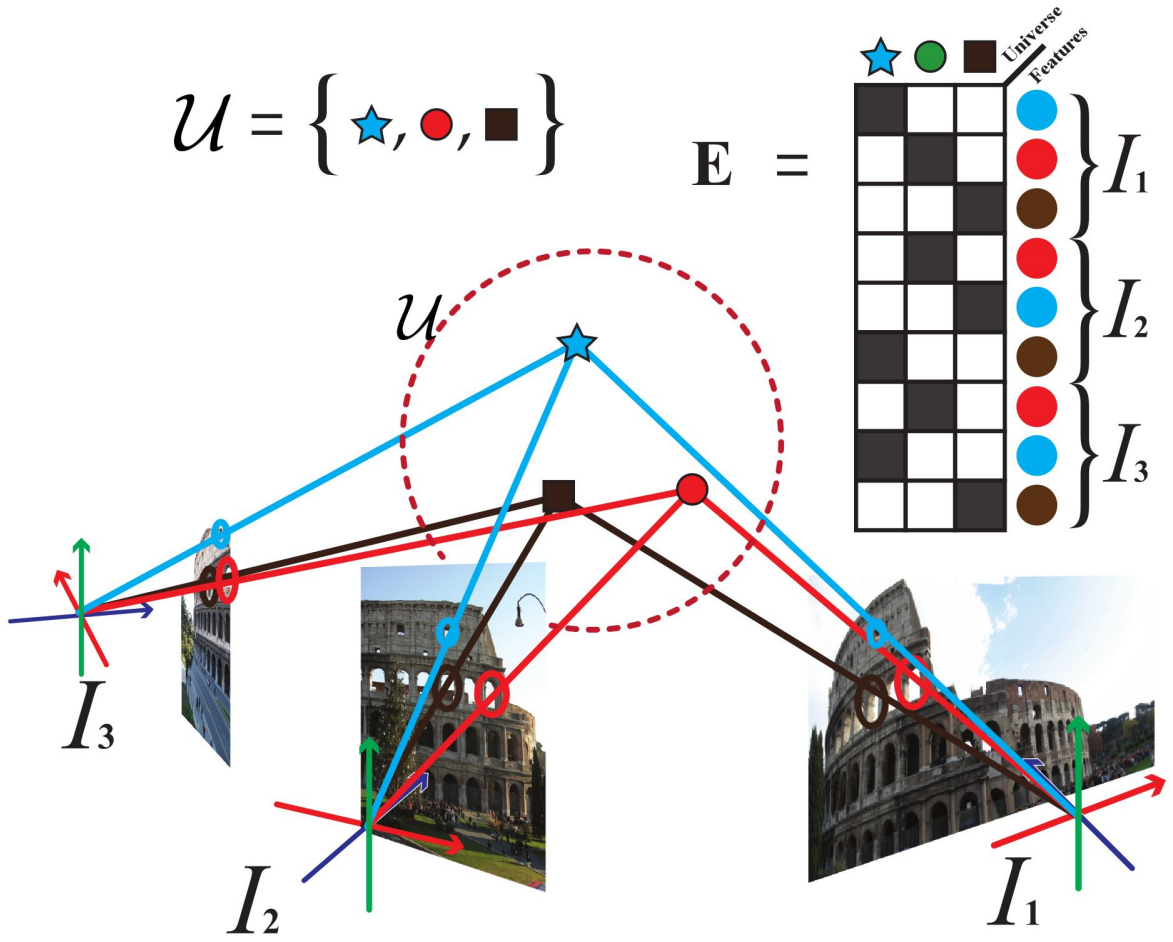
Graph Convolutional Networks

- Operates in applying successive Graph operations, specifically applying the Laplacian to node embeddings
  - Equivalent to weighted average of neighbors
  - Apply linearity before Laplacian, and non-linearity afterwards

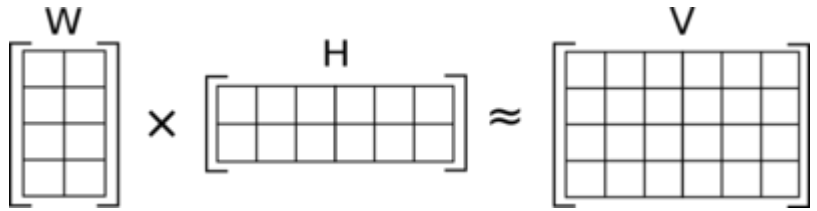


Cycle Consistency

- Finding inconsistencies in cycles of matches in the images gives an error metric
- Traditionally solved using Matrix Factorization on correspondence matrix
- We use equivalent to low rank matrix factorization with a Neural Network as loss

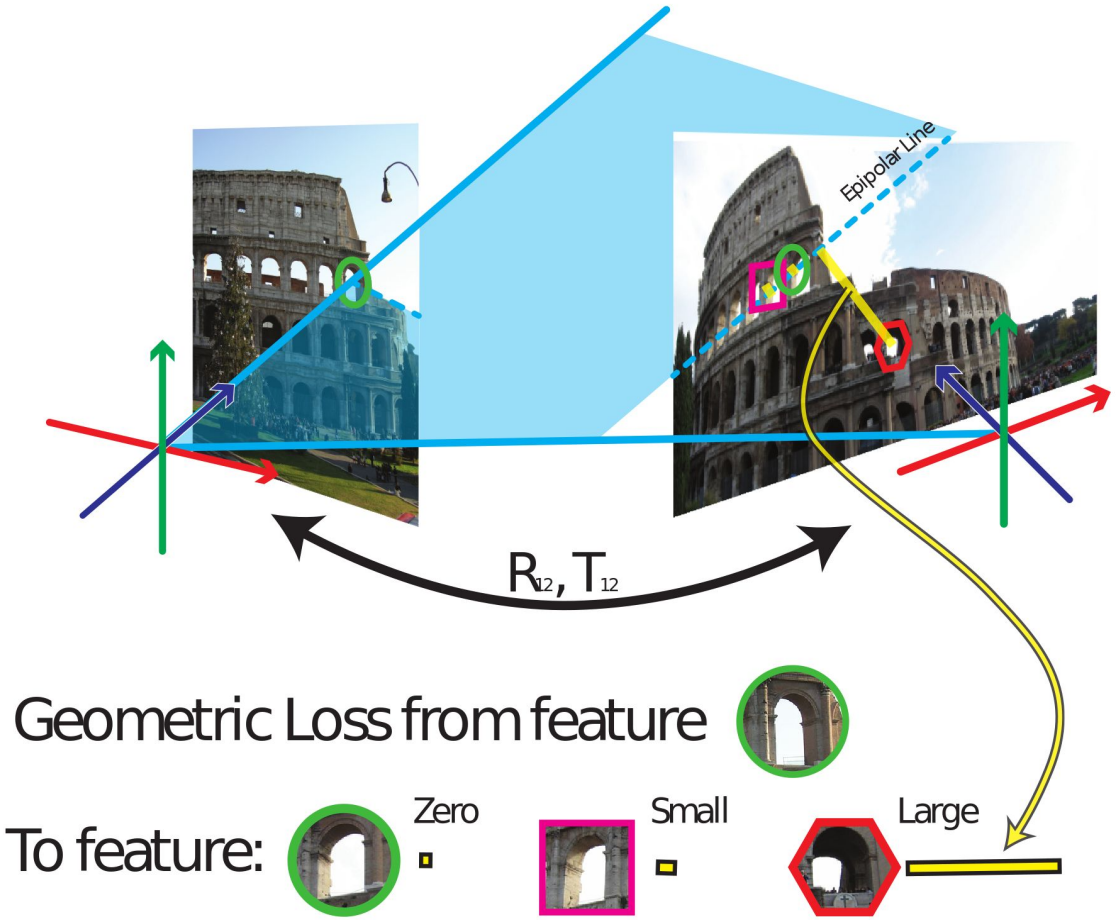


$$\|EE^T - \tilde{X}\|^2 = \|X_n X_n^T - \tilde{X}\|^2$$

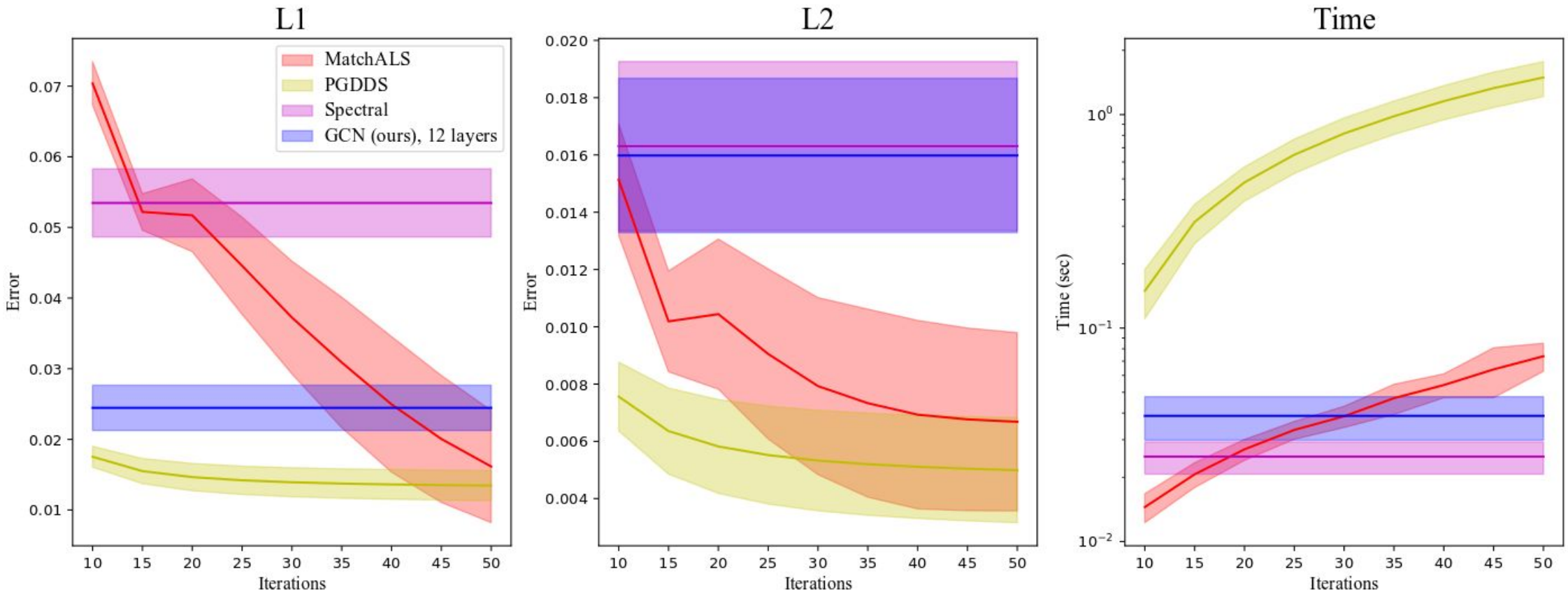


Advantages of this approach

- As mentioned, unsupervised approach
- Can add additional losses in training for geometric consistency without needing them at testing time
- Graph Neural Net formulation makes parallelization between images trivial at test time
- Limitations: Graph Neural Networks are much less expressive compared to their CNN counterparts
  - Working on attention mechanisms to help expressivity more



GCN vs Optimization - Rome 16K

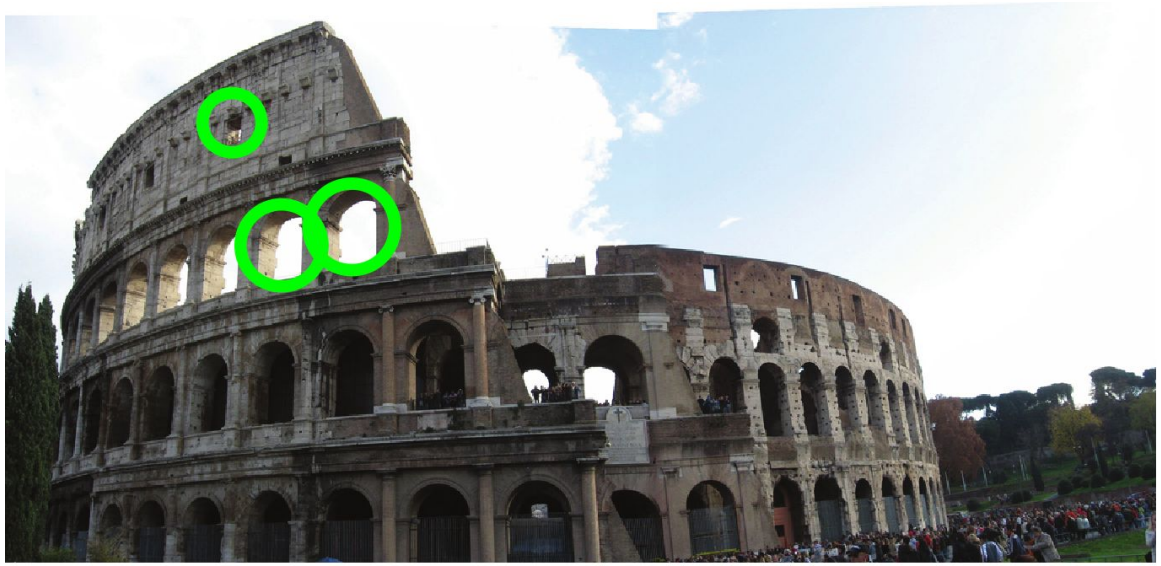


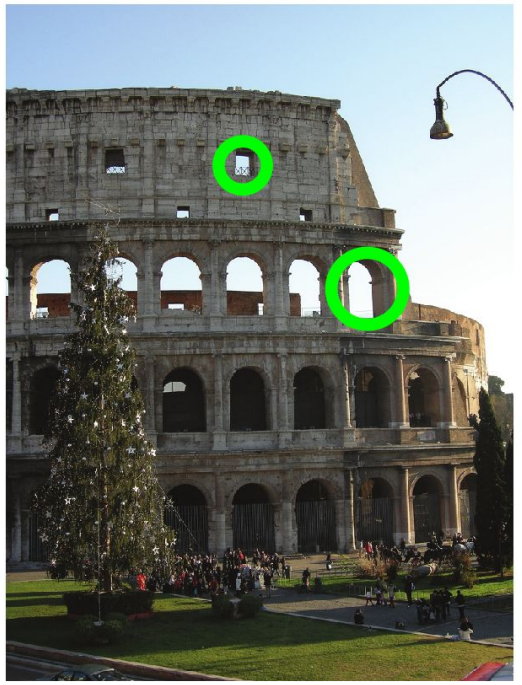
All Graphs Lead to Rome: Learning Geometric and Cycle-Consistent Representations with Graph Convolutional Networks

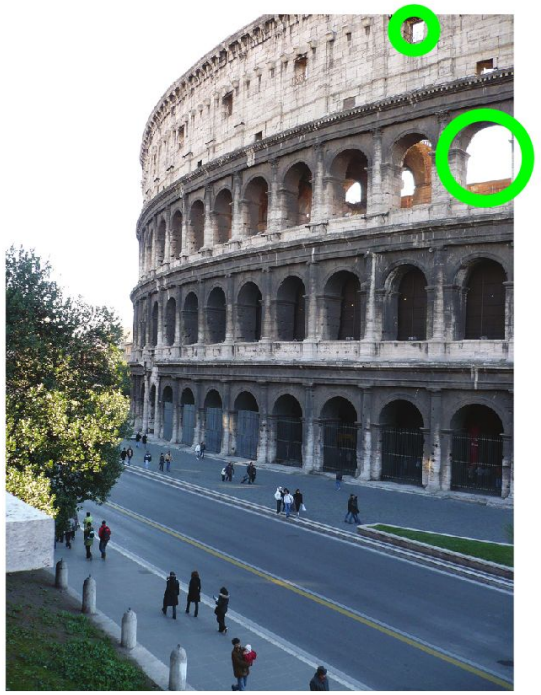
Stephen Phillips
Kostas Daniilidis

Motivation

- Finding correspondence between radically different viewpoints is difficult
- Hand-crafted local appearance models don’t work across such large variations
- Machine Learning shows promise in creating representations that can handle such variations

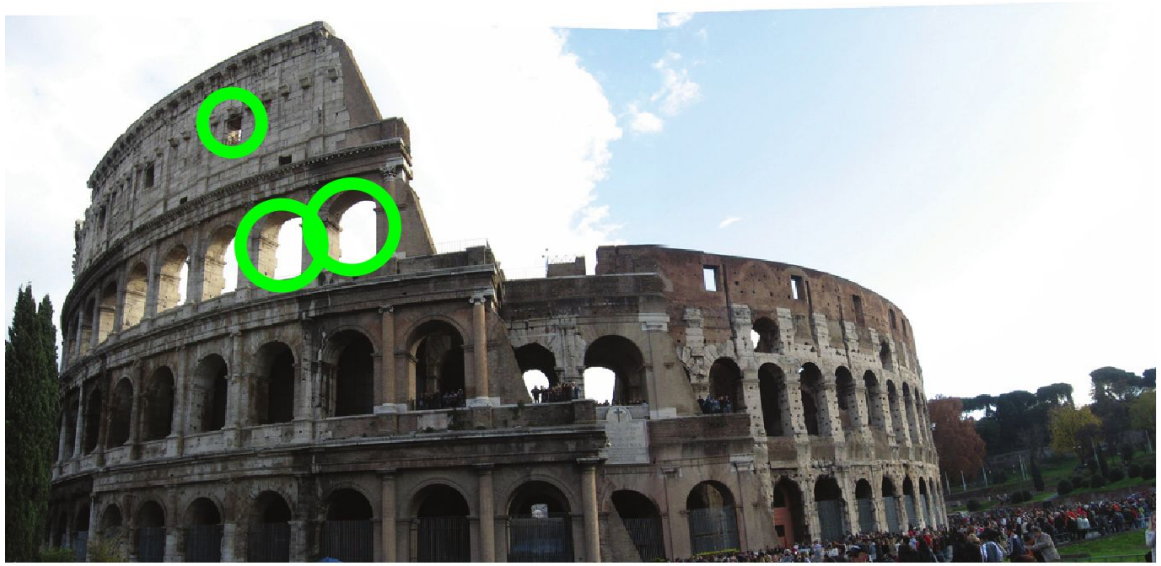


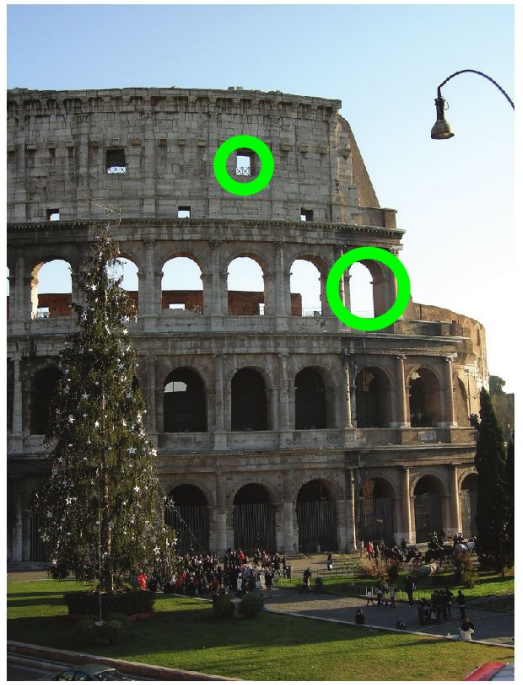


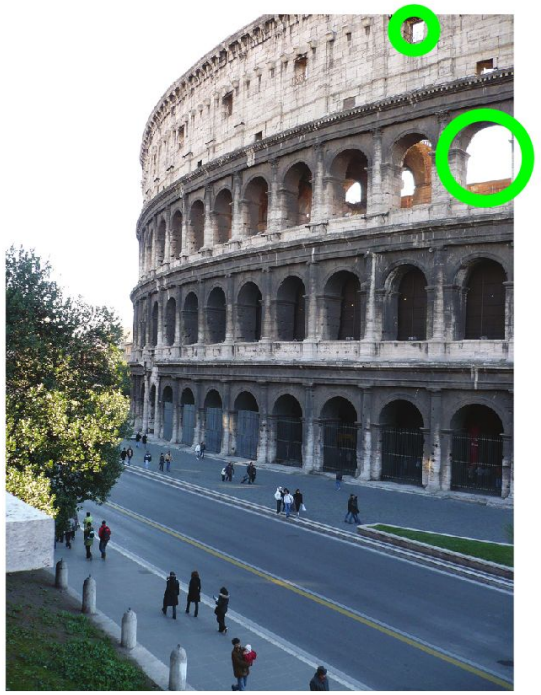


Motivation

- Correspondence is a fundamental problem in Computer Vision and navigation
- Recently Machine Learning techniques have performed very well for many Computer Vision tasks
- Hope to bring the advantages of ML techniques to this domain
- Want to find novel representations by enforcing geometric and matching constraints



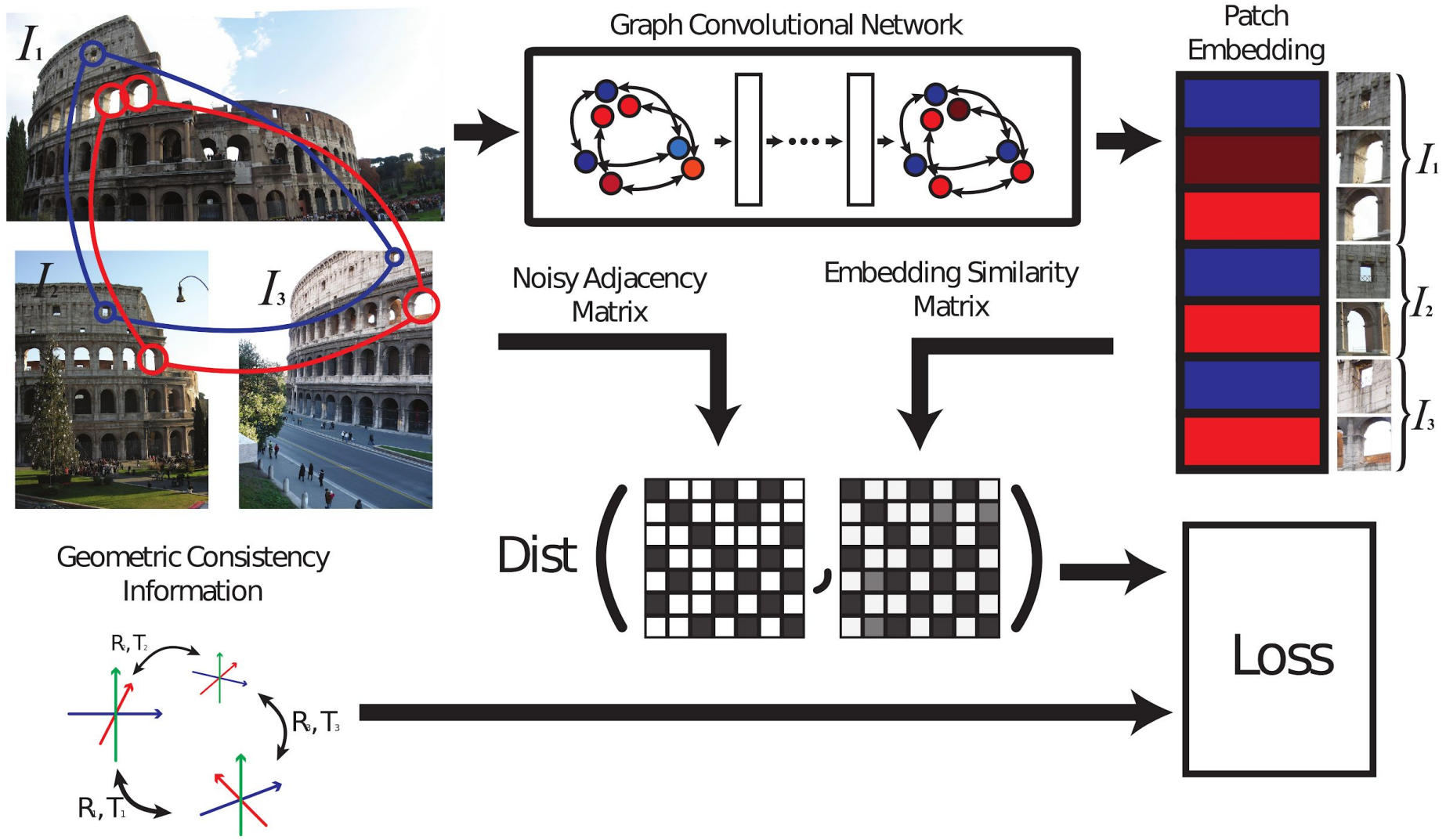




Challenges

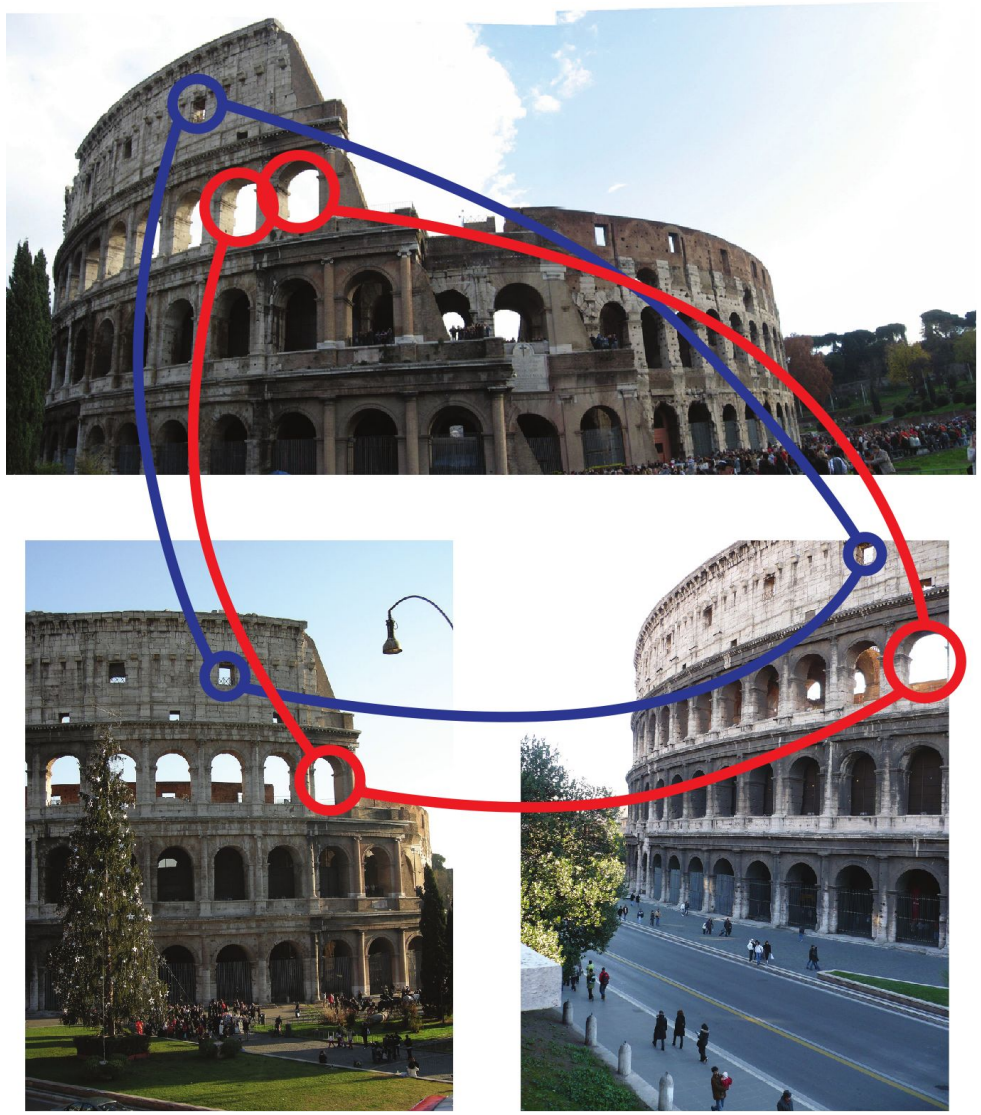
- How to represent sparse correspondence in a ML framework
- Lack of labelled data
- Training end to end to learn better features
- Additional: Distributed challenges? Incorporating geometric information in training?

Outline


I_1
I_2
I_3
Graph Convolutional Network
Patch Embedding
I_1
I_2
I_3
Noisy Adjacency Matrix
Embedding Similarity Matrix
Dist
Loss
Geometric Consistency Information
R, T
R, T
R, T

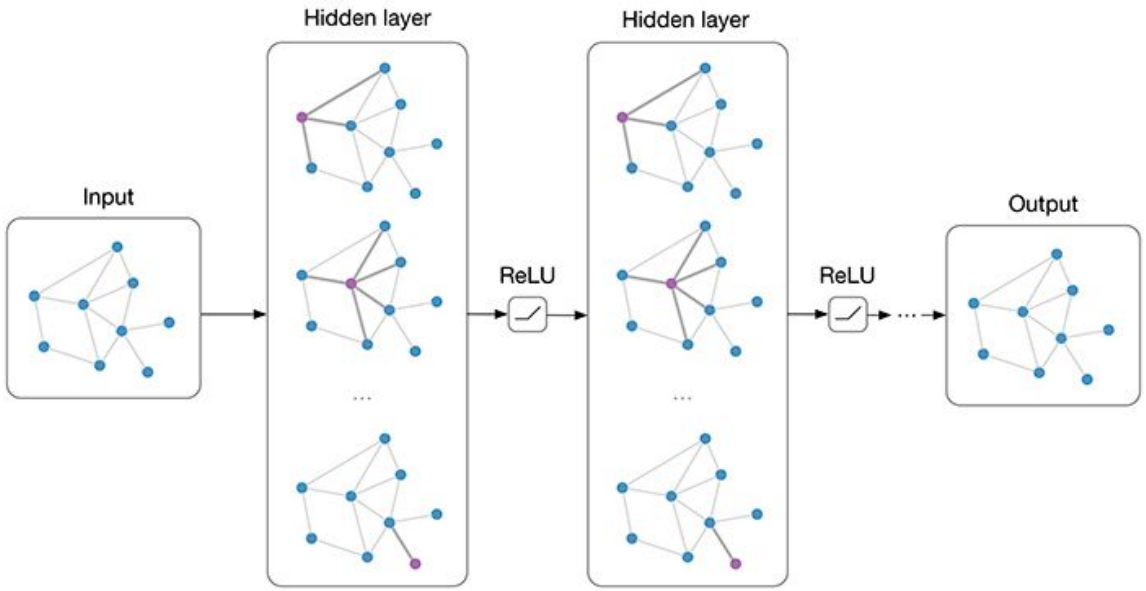
Correspondences as a Graph

- Can treat a correspondences as a graph
  - Each Node represents a point on the image, with possibly a feature attached to it
  - Each edge represents a putative correspondence
  - Potential errors in the correspondence



Graph Neural Networks

- Have embedding on each node, propagate embeddings using the graph
- Can use to operate on the graph of putative correspondences



Graph Neural Networks

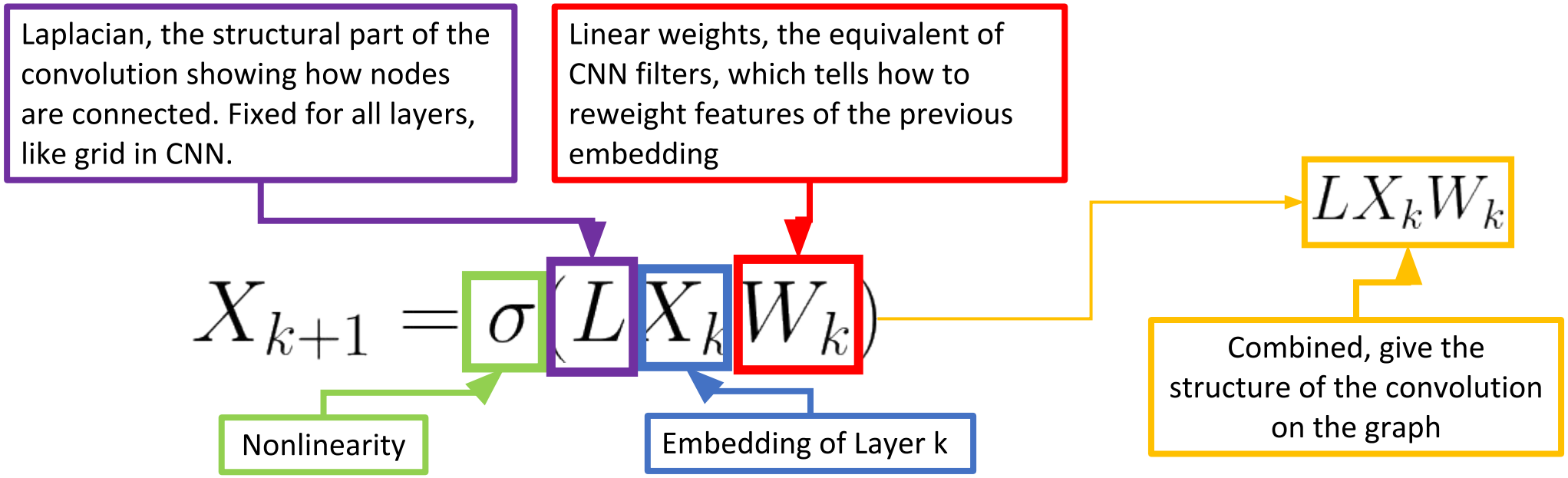
- Graphs are challenging to define operations on
- Standard to represent the graph by its Laplacian, a function of its adjacency matrix:

$$L = D - A$$

- If graph structure known in advance, can use the eigenvectors of Laplacian for convolutional kernels (e.g. Spectral Nets, Bruna et al 2014)
- If structure not known in advance other methods have to be used e.g. Message Passing NNs (Gilmer et al 2017) or what we use Graph Convolutional Nets (GCNs) (Kipf and Welling 2017)

Graph Convolutional Networks

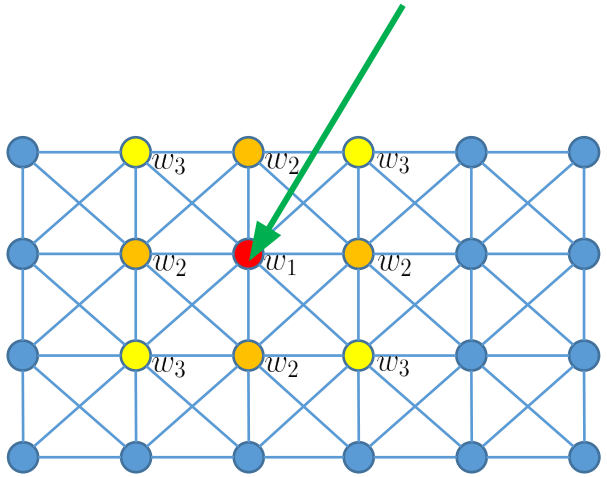
- Operates in applying successive Graph operations, specifically applying the Laplacian to node embeddings
  - Equivalent to weighted average of neighbors
  - Apply linearity before Laplacian, and non-linearity afterwards



Graph Convolution Example

- Look at the simple case of graph of regular grid
- Equivalent to radially symmetric convolutional filters in this case

Weights around this node



$(L \otimes W_k)$

L

$X_k W_k$

Vectorize

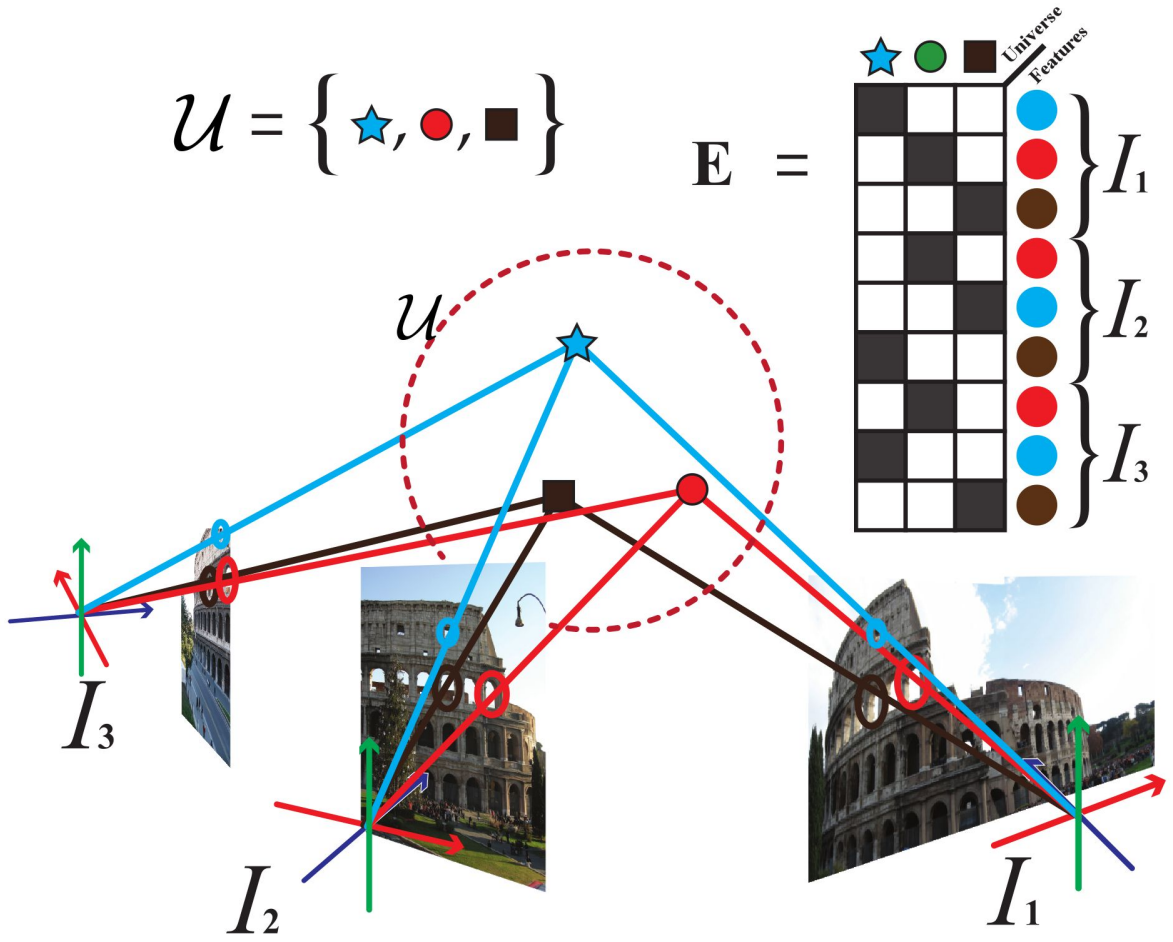
X_k

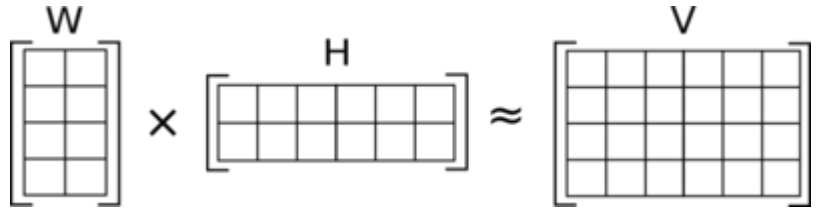
Laplacian is just weighted convolution pattern

Repeated convolution patterns for each feature dimension

Cycle Consistency

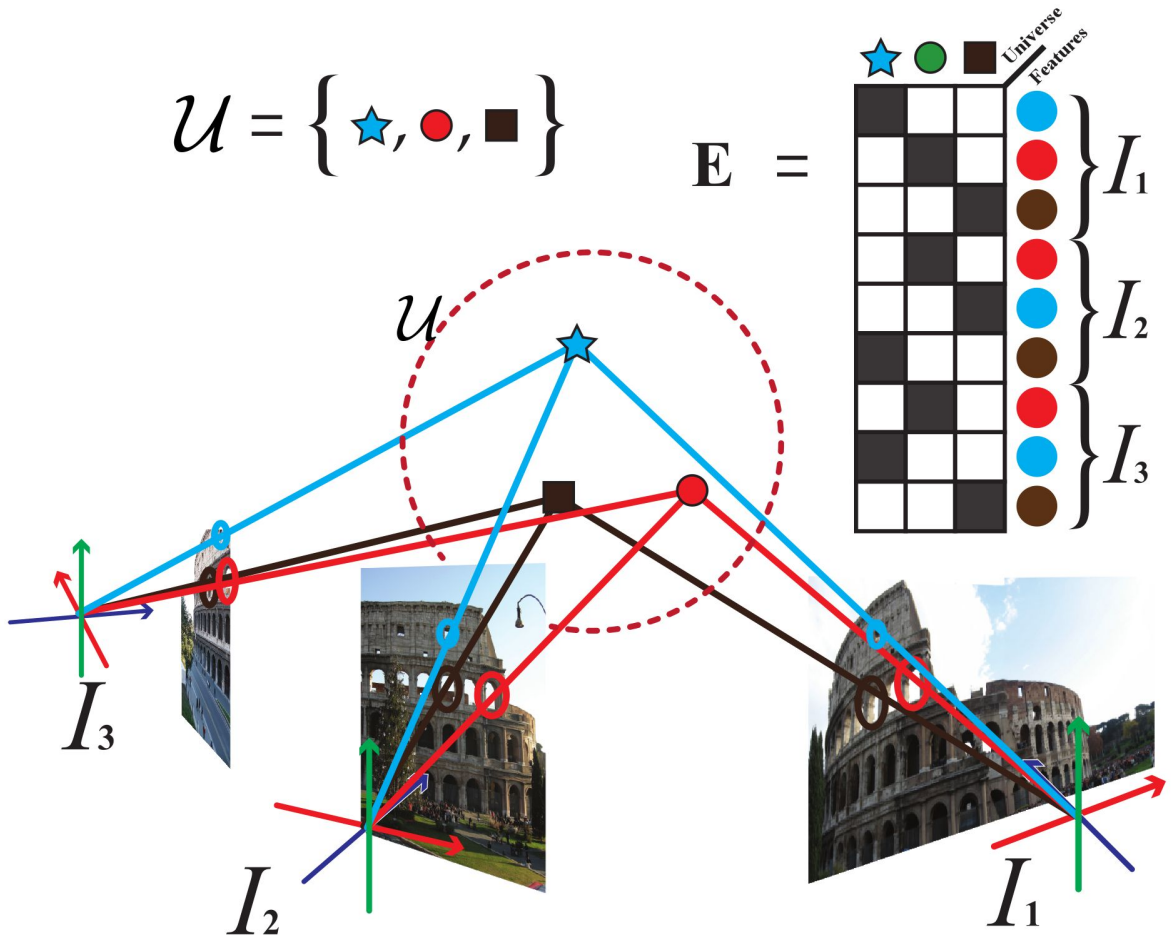
- Given set of images with pairwise correspondences - might be errors
- Finding inconsistencies in cycles of matches in the images gives an error metric
- Traditionally solved using Matrix Factorization on correspondence matrix





Cycle Consistency as Loss

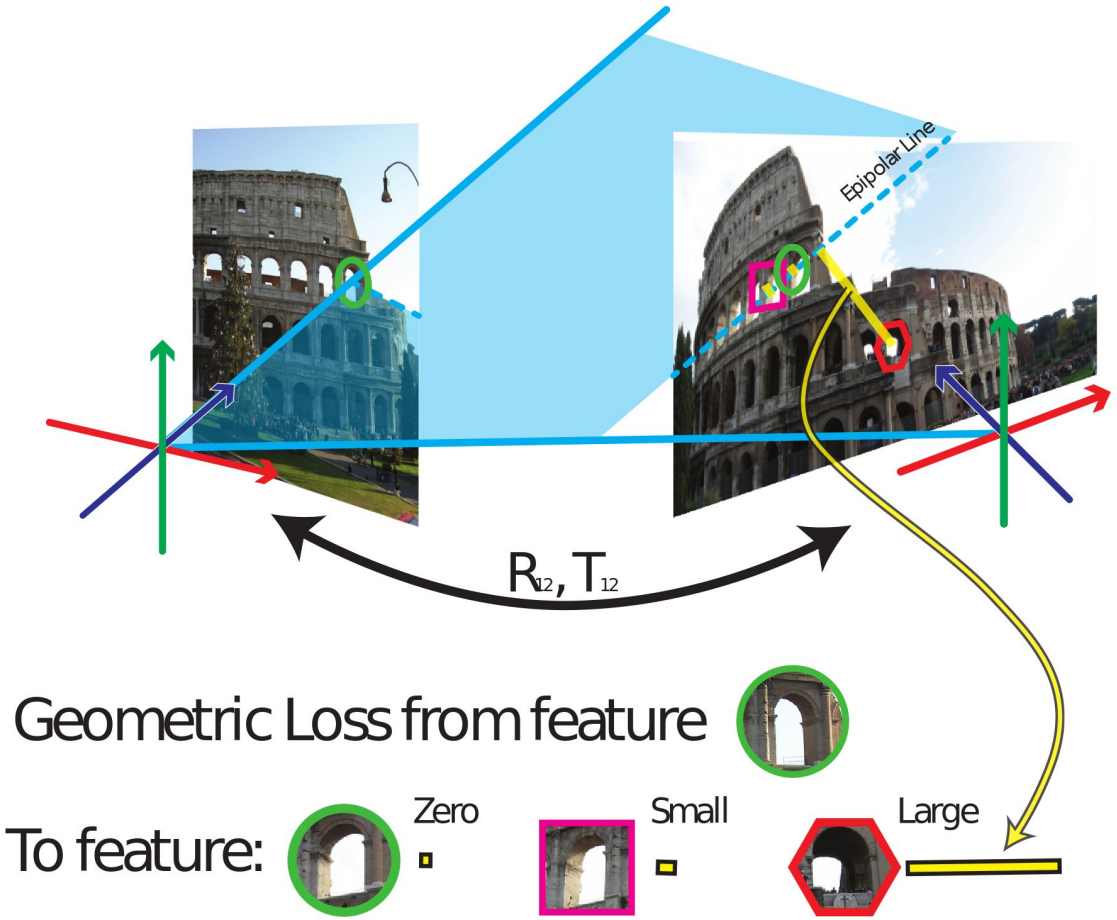
- Can use Cycle Consistency as loss on similarity between learned features across images
- Just need feature matches, don't need ground truth – semi-supervised!
- This makes it equivalent to low rank matrix factorization with a Neural Network



$$\|EE^T - \tilde{X}\|^2 = \|X_n X_n^T - \tilde{X}\|^2$$

Advantages of this approach

- As mentioned, unsupervised approach
- Can add additional losses in training for geometric consistency without needing them at testing time
- Graph Neural Net formulation makes parallelization between images trivial at test time
- Limitations: Graph Neural Networks are much less expressive compared to their CNN counterparts
  - Working on attention mechanisms to help expressivity more



Synthetic Data Experiments

- Data generation: Generate synthetic graph based on synthetic images, add noise model, extract adjacency matrix

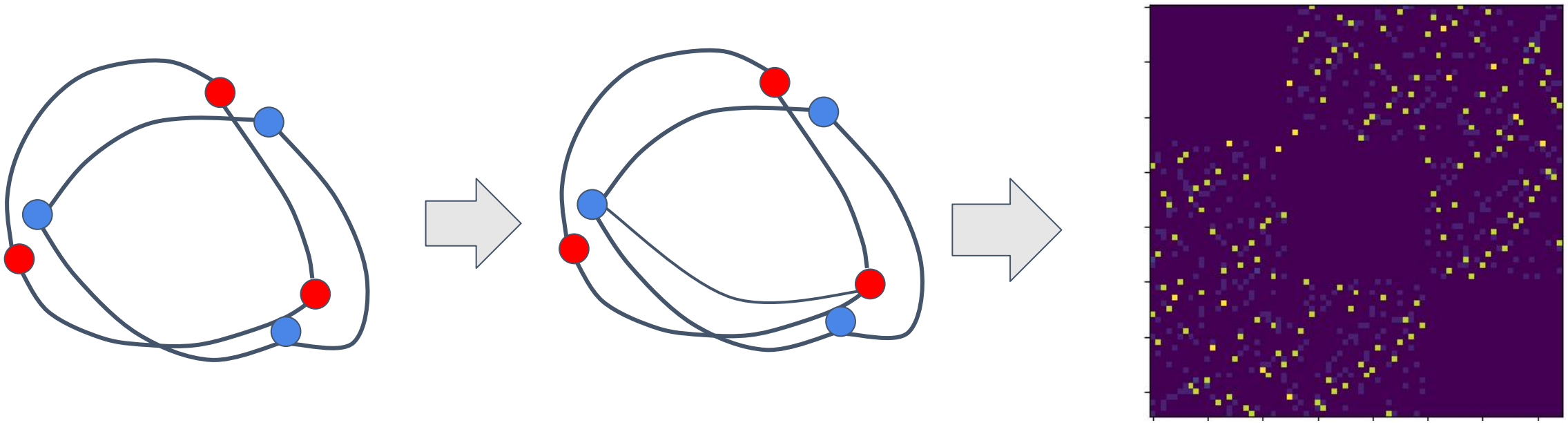


Table of Results

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

Real World Data

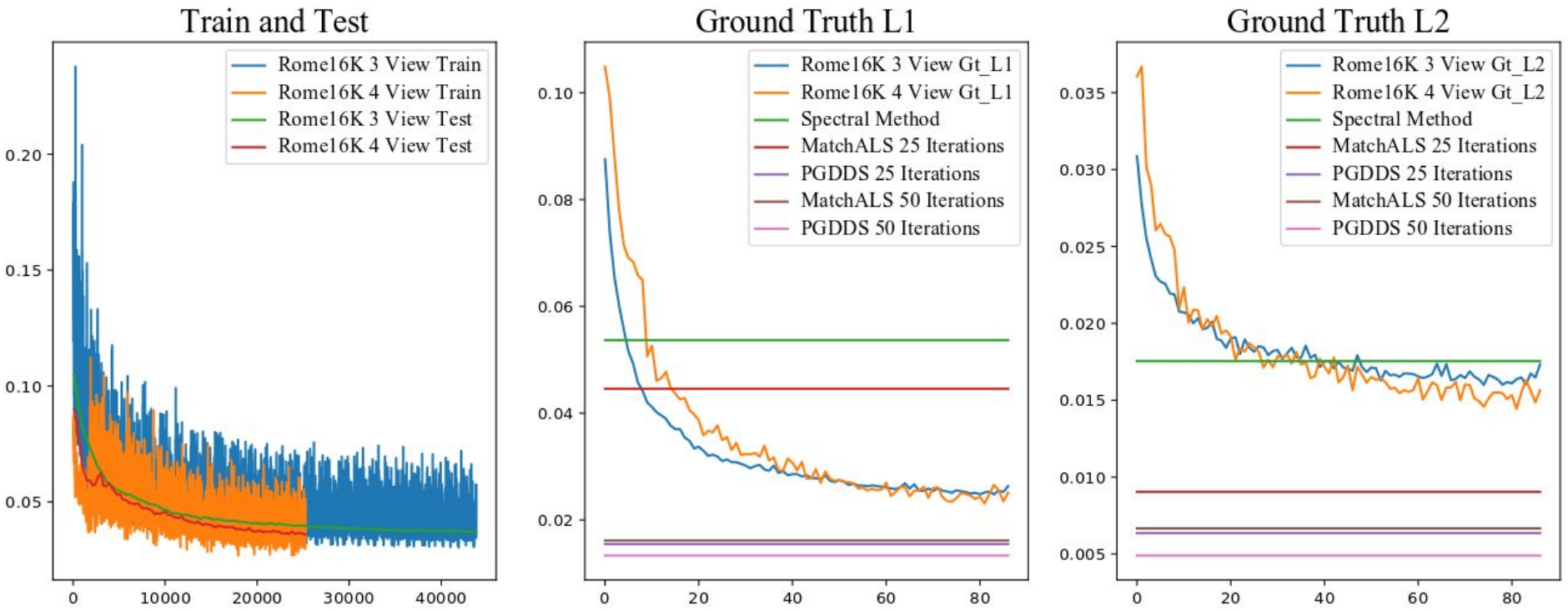
Rome 16K

Triplets

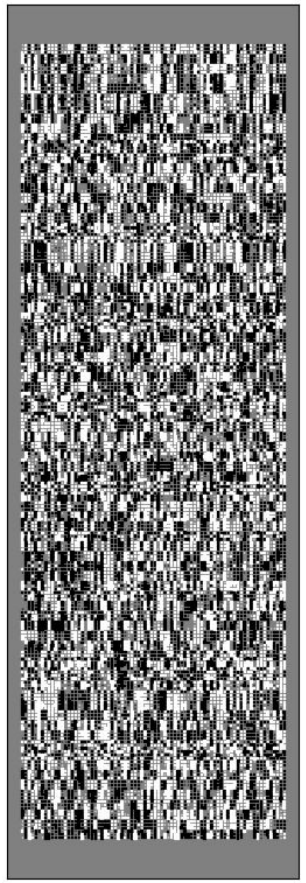
80 Features selected per image

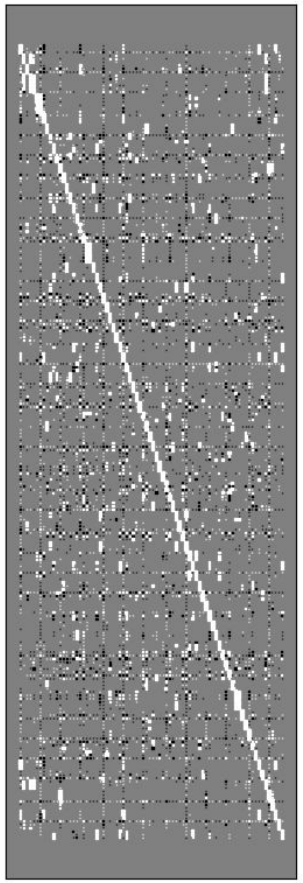
1M+ Examples

Rome16K Train/Test Curves



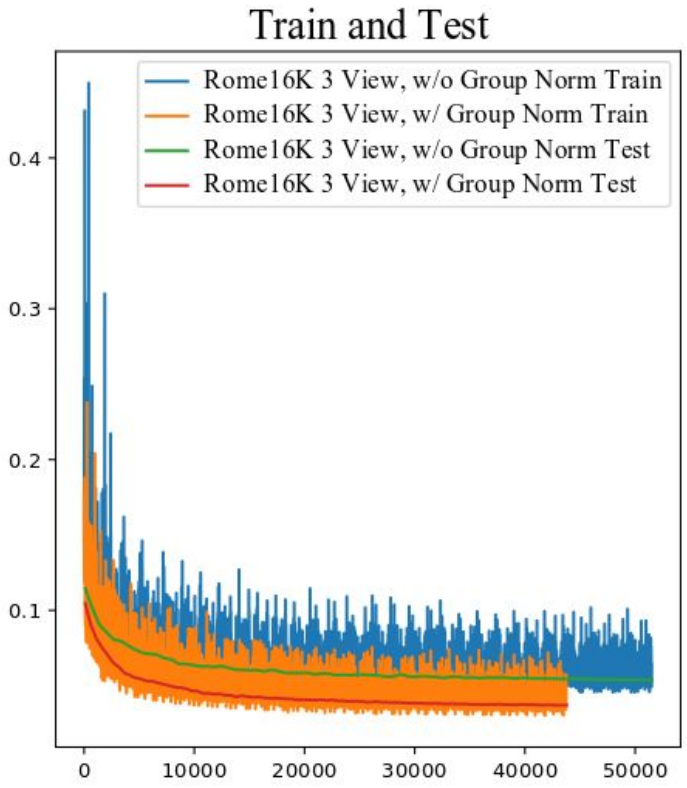
Rome16K Embeddings

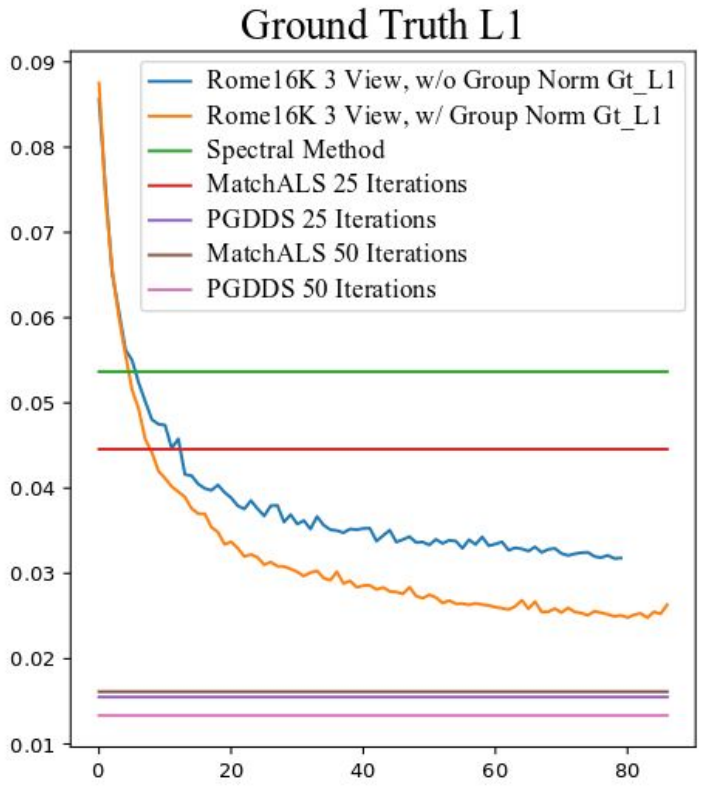


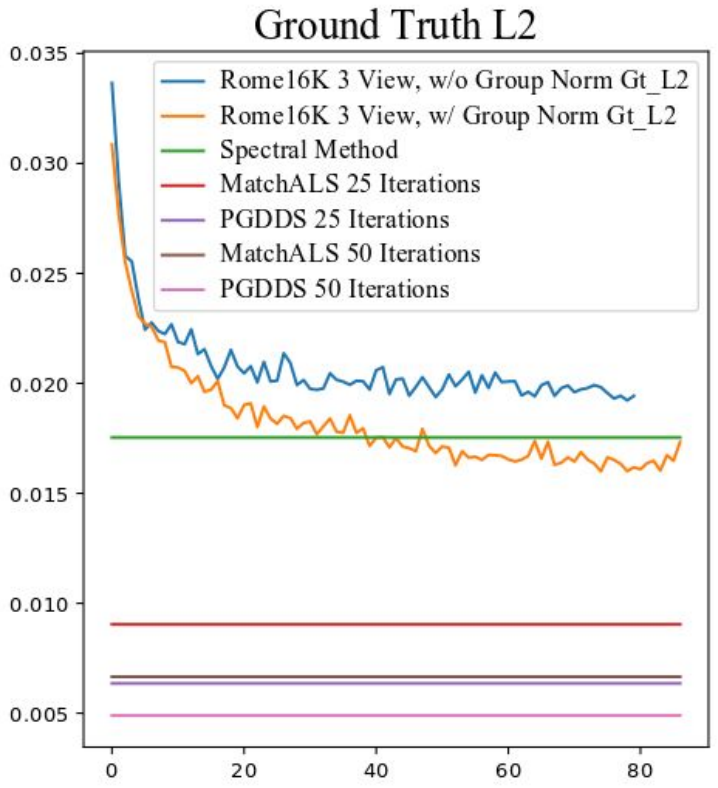


Group Norm

- Trick to make this work - makes it not parallelizable. For future work



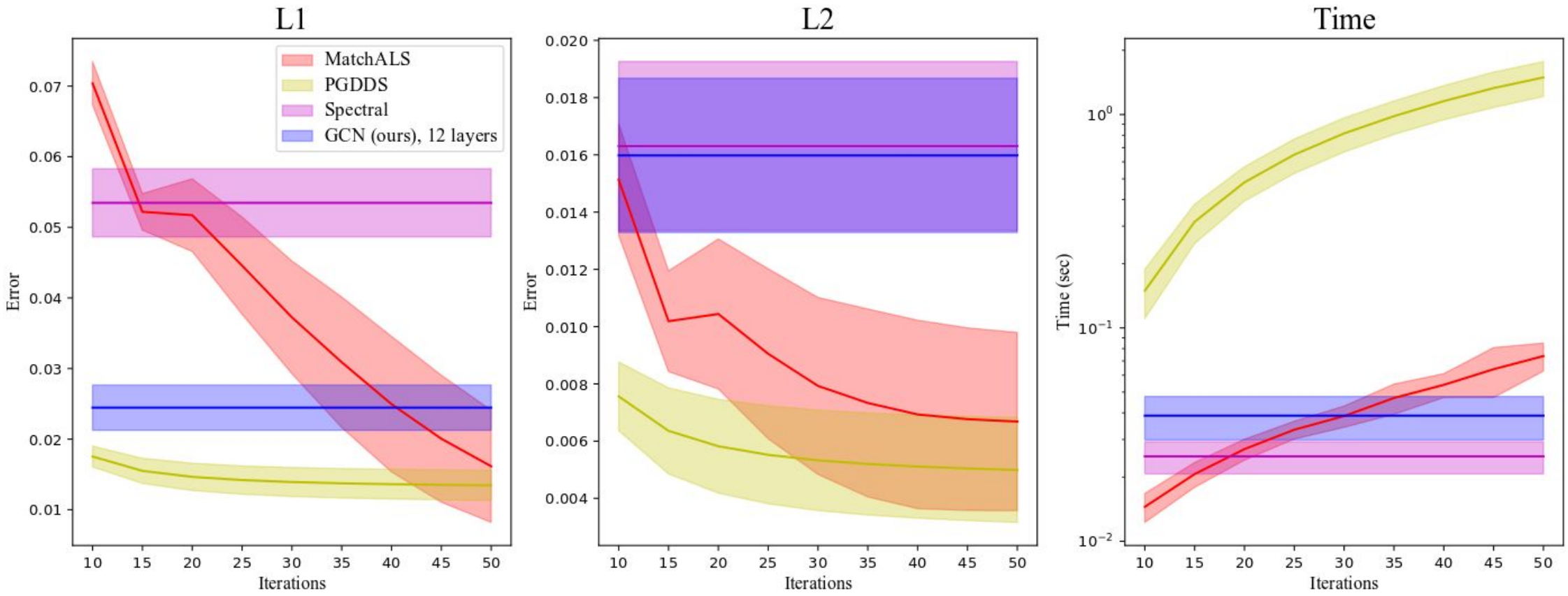




Method (3 Views)	L_1 Loss	L_2 Loss	Run Time (sec)
MatchALS [31] 15 Iterations	0.052 ± 0.003	0.010 ± 0.002	0.021 ± 0.003
MatchALS [31] 25 Iterations	0.045 ± 0.007	0.009 ± 0.003	0.034 ± 0.003
MatchALS [31] 50 Iterations	0.016 ± 0.008	0.007 ± 0.003	0.065 ± 0.006
PGDDS [17] 15 Iterations	0.016 ± 0.002	0.006 ± 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)	$\mathbf{0.025 \pm 0.003}$	$\mathbf{0.016 \pm 0.003}$	$\mathbf{0.039 \pm 0.009}$
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)	$\mathbf{0.023 \pm 0.003}$	$\mathbf{0.015 \pm 0.002}$	$\mathbf{0.056 \pm 0.017}$

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.

GCN vs Optimization Iterations



Conclusion

- We can learn to refine multi-image correspondences using a Graph Neural Network
- We perform comparably to optimization based baselines with fewer layers
- Future work:
  - Using more sophisticated graph convolutional network structures.
  - Making this distributed
  - Training image features along with the multi-image matching in an end-to-end fashion