Compressive Recovery of Signals Defined on Perturbed Graphs

Anonymous authors
Paper under double-blind review

Abstract

Recovery of signals with elements defined on the nodes of a graph, from compressive measurements is an important problem, which can arise in various domains such as sensor networks, image reconstruction and group testing. In some scenarios, the graph may not be accurately known, and there may exist a few edge additions or deletions relative to a ground truth graph. Such perturbations, even if small in number, significantly affect the Graph Fourier Transform (GFT). This impedes recovery of signals which may have sparse representations in the GFT bases of the ground truth graph. We present an algorithm which simultaneously recovers the signal from the compressive measurements and also corrects the graph perturbations. We analyze some important theoretical properties of the algorithm. Our approach to correction for graph perturbations is based on model selection techniques such as cross-validation in compressed sensing. We validate our algorithm on signals which have a sparse representation in the GFT bases of many commonly used graphs in the network science literature. An application to compressive image reconstruction is also presented, where graph perturbations are modeled as undesirable graph edges linking pixels with significant intensity difference. In all experiments, our algorithm clearly outperforms baseline techniques which either ignore the perturbations or use first order approximations to the perturbations in the GFT bases.

1 Introduction

Efficient acquisition of data arranged on the nodes of a graph (i.e. a graph signal) may be performed via the technique of Compressed Sensing (CS) Candes & Wakin (2008). Instead of recording the data at each node of the graph separately, a small number of random linear measurements of the graph signal may be acquired in a domain-specific manner. CS techniques allow accurate recovery of a high-dimensional vector from a small number of linear measurements, given sufficient conditions such as sparsity/compressibility of the signal representation in a known orthonormal basis, and measurement matrix characteristics such as the restricted isometry property (RIP) Candes & Wakin (2008). In some applications, graph signals defined on the nodes of an undirected, unweighted graph, possess a sparse representation in terms of the eigenvectors of the Laplacian matrix of the graph, which are also known as a Graph Fourier Transform (GFT) basis for such graphs in the Graph Signal Processing (GSP) literature Ortega et al. (2018). Given the sparsity of this representation, such graph signals may be recovered from the compressive linear measurements using CS decoding algorithms. Compressive graph signal measurements naturally arise in sensor networks Zhu & Rabbat (2012), group testing with side information such as contact tracing graphs Goenka et al. (2021), or in image acquisition if the image is regarded as a graph.

In this work, we consider the case when a graph signal has been compressively acquired, but there is some uncertainty in the knowledge of the graph on which the signal was defined. A *nominal* graph is known, which has the same nodes as the *actual* (unknown) graph, but has a small number of edge perturbations (edge additions and/or deletions) relative to the actual graph. Due to this, significant uncertainty is induced in the GFT basis of the graph, since the perturbation of a few edges of the graph will perturb its Laplacian matrix, and correspondingly its eigenvectors. Hence the graph signal will not be recovered accurately by a CS decoding algorithm if the GFT basis of the nominal graph is used, and some alternative approach is

needed for recovery of the graph signal. In some applications, it may be desirable to recover the actual graph as well. We refer to the problem of recovery of the actual graph and the graph signal from compressive measurements as the *Compressive Perturbed Graph Recovery* (CPGR) problem.

We present a method called Greedy Edge Selection (GES), which solves this problem by refining the nominal graph one edge at a time, based on the cross-validation (CV) errors of the signals recovered using these graphs on a held-out set of measurements. The algorithm keeps proceeding as long as the CV errors of the successive graphs keep decreasing. Finally, the algorithm outputs the refined graph and the graph signal defined using the GFT basis of the refined graph. To summarize, our main contributions in this work are:

1. We present the novel (to our best knowledge) problem of Compressive Perturbed Graph Recovery (CPGR), in which there is uncertainty in the orthonormal basis used for CS recovery, due to edge perturbations in the underlying graph.

2. We present the Greedy Edge Selection (GES) method (Sec. 3.2) for solving the CPGR problem, using an approach based on CV errors.

3. We frame the problem of edge-aware recovery of patch-wise compressively acquired images (such as in Kulkarni et al. (2016)) as a CPGR problem. As an extension of GES to this special case, we propose Inferred Linear-Edge Compressive Image Recovery (ILECIR, Sec. 4), an algorithm which recovers images simultaneously with linear image edges in the patches of the image via structured/coupled perturbation of the edges of a 2D lattice graph (unlike unstructured edge perturbations performed in GES).

4. Using CV theory for compressed sensing Zhang et al. (2014), we prove high-probability signal and graph recovery guarantees for a brute-force version of our algorithms (called 'Brute Force Graph Selection' or BFGS). We also provide similar guarantees for solution improvement at each step of GES, and also in the ILECIR algorithm.

5. We empirically validate the GES algorithm on signals on a variety of graphs commonly used in the Network Science literature net (2024). We also perform simulations on interpolation of signals defined on perturbed graphs. Likewise, we validate the ILECIR algorithm on 40 images of various kinds – natural images, synthetically generated piece-wise smooth images, cartoon images, and depth-maps of indoor scenes (Sec. 5). Our algorithms outperform the baseline method of using the GFT basis of the nominal graph or the GFT basis produced by first-order perturbations, in a standard CS decoding algorithm.

1.1 Applications of CPGR

The CPGR problem is motivated by the following applications (see also Sec. 2 after equation 1):

1. **Compressive Image Recovery:** 2D Images can be regarded as signals defined on the nodes of a grid graph. The 2D DCT is one of the possible GFT bases for such a graph. However, GSP principles allow this to be generalized to compressive reconstruction of images defined on any other type of pixel grid, such as hexagonal Middleton & Sivaswamy (2005), or on the vertices of a 3D mesh Lalos et al. (2016), for which the 2D DCT is no longer the appropriate GFT basis. In all such cases, the nominal graph is regarded as the perturbed version of the actual graph which allows sparser image representation. The graph edges (in the nominal graph) to be dropped correspond to edges between graph nodes that lie on two different sides of an image edge or object-object boundary. See Sec. 4 for more details.

2. **Graph Signal Interpolation:** A second motivating example is the case where the data values at only a subset of m out of n graph nodes ($m < n$) are available, and the remaining are missing and need to be inferred. This situation comes up in mass testing in a pandemic, where it is impossible to test each person, and contact tracing information is useful for inferring the health status of individuals who have not been tested. In this case, note that the sensing matrix is a randomly

chosen row-subset of the identity matrix (depending on the particular nodes for which signal values were observed). Each person (or a group of individuals such as a family) is regarded as a node of the CT graph. The infection status at the node is the signal value, and CT graph edges are created between different persons who are in contact with each other or between different groups of people which regularly intermingle. If we assume a diffusion model for the spread of the infection levels across people who come in contact, we can model the vector of infection levels as being sparse or compressible in the GFT of the CT graph. But CT graphs are known to contain errors due to various limitations of Bluetooth or other modalities for acquiring CT information Kleinman & Merkel (2020). See Sec. 5.1 under 'Signal Interpolation on Perturbed Graphs' for more details.

3. **Pooled Testing using Contact Tracing Graphs:** Consider the case of pooled testing in a pandemic situation Ghosh et al. (2021), using CT graphs Goenka et al. (2021). Pooled testing involves replacing individual tests on n different individuals by tests on $m \ll n$ pools, where each pool is creating by mixing small, fixed-volume portions of the samples of a subset (containing $r < n$ individuals) of the n individuals. The participation of the different individuals in different pools is encoded by a binary 'pooling matrix'. The aim is to infer the infection levels of the n individuals given knowledge of the pooling matrix and the results on the m pools. This can be framed as a CS problem as in Ghosh et al. (2021). Successful estimation of infection levels requires the n-element vector of individual infection levels to be sparse, and also requires the pooling matrix to obey certain properties Ghosh et al. (2021). However additional information is available in CT graphs, which can be used to improve the inference for the same m, assuming a diffusion model for infection spread. But CT graphs can contain errors as mentioned previously, and hence this is another important application of our theory.

Organization of the paper: The remainder of this paper is organised as follows: Sec. 2 defines the CPGR problem formally and gives an overview of the literature tackling similar problems. Sec. 3 and 4 describe the methods presented in this paper in detail. Several numerical results for our methods are presented in Sec. 5. Finally, conclusions and future work are discussed in Sec. 6. A **glossary** containing various symbols (variables and algorithm names) used in the main paper is presented in Table S.V of the supplemental material.

2 Problem Statement

Consider that we are given linear, and possibly noisy, compressive measurements $\boldsymbol{y} \in \mathbb{R}^m$ of an unknown graph signal defined on the vertices of an undirected, unweighted graph $\mathcal{G}_{\text{actual}} := (\mathcal{V}, \mathcal{E}_{\text{actual}})$ where the set of the nodes is denoted by $\mathcal{V} \triangleq \{1, \ldots, n\}$ (and thus $|\mathcal{V}| = n$), and the set of edges is denoted by $\mathcal{E}_{\text{actual}}$. The graph signal is given by a vector $\boldsymbol{x}^* \in \mathbb{R}^n$ with each entry representing the value on the corresponding graph node. Hence we have

$$\boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{x}^* + \boldsymbol{\eta}, \tag{1}$$

where $\boldsymbol{\Phi}$ is a $m \times n$ 'measurement/sensing matrix' with $m \ll n$ (since we are in the compressive regime), and $\boldsymbol{\eta}$ is a noise vector, each of whose entries is assumed to be i.i.d. Gaussian with mean 0 and variance σ^2. In our setting, the graph $\mathcal{G}_{\text{actual}}$ is not known with full accuracy. Instead, we have access to the graph $\mathcal{G}_{\text{nominal}} = (\mathcal{V}, \mathcal{E}_{\text{nominal}})$, which has the same nodes as $\mathcal{G}_{\text{actual}}$, but the set of edges $\mathcal{E}_{\text{nominal}}$ contains a few edge perturbations (additions or deletions) relative to $\mathcal{E}_{\text{actual}}$. We now give a few examples of actual and nominal graphs in practical applications, referring to Sec. 1.1. In compressive image recovery, the nominal graph is the grid graph in which each node is a pixel and is connected by graph edges to its four neighbors, whereas the actual (unknown) graph is the one in which certain graph edges (corresponding to image edges) are dropped. In graph signal interpolation or pooled testing, the nominal graph is the available CT graph obtained via Bluetooth, whereas the actual graph is the CT graph in which each graph edge represents an actual contact between two individuals (irrespective of whether or not it was recorded by Bluetooth).

We assume that an upper bound d_0 on the number of perturbations is known, i.e. $|\mathcal{E}_{\text{actual}} - \mathcal{E}_{\text{nominal}}| + |\mathcal{E}_{\text{nominal}} - \mathcal{E}_{\text{actual}}| \leq d_0$. For any undirected graph with adjacency matrix $\boldsymbol{W}$, its Laplacian matrix is given by $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$, where $\boldsymbol{D}$ is the diagonal matrix whose i^{th} diagonal entry contains the degree of the i^{th} node.

For simplicity, we assume unweighted graphs. For undirected graphs, the Laplacian matrix is positive semi-definite and has real-valued eigenvectors and eigenvalues. Let $\boldsymbol{L}_{\text{actual}}$ be the (unknown) Laplacian matrix of $\mathcal{G}_{\text{actual}}$, and let $\boldsymbol{L}_{\text{actual}} = \boldsymbol{V}_{\text{actual}}\boldsymbol{\Lambda}_{\text{actual}}\boldsymbol{V}_{\text{actual}}^T$ be its eigen-decomposition. From the GSP literature, we know that the eigenvectors $\boldsymbol{V}_{\text{actual}}$ form a GFT basis, with each eigenvector having an associated 'frequency' – the eigenvalue – which is a measure of its variation across the neighbouring nodes of the graph $\mathcal{G}_{\text{actual}}$ (Ortega et al., 2018, Sec. II.E). We assume that $\boldsymbol{x}^*$ is *sparse-spectrum*, i.e. its GFT (Ortega et al., 2018, Eqn. 14) given by $\boldsymbol{\theta}^* = \boldsymbol{V}_{\text{actual}}^T\boldsymbol{x}^*$ has very few non-zero entries. We also assume that $\mathcal{G}_{\text{nominal}}$ and $\mathcal{G}_{\text{actual}}$ have distinct eigenvalues. The case of repeated eigenvalues is discussed later in this section. The goal is to recover the original signal $\boldsymbol{x}^*$, as well as the actual graph $\mathcal{G}_{\text{actual}}$ given just $\boldsymbol{y}, \boldsymbol{\Phi}$ and $\mathcal{G}_{\text{nominal}}$. This is formally defined as follows: [Compressive Perturbed Graph Recovery (CPGR) Problem] Given $\boldsymbol{y}$, $\boldsymbol{\Phi}$, $\mathcal{E}_{\text{nominal}}$ and d_0, find $\boldsymbol{x}^*$ and $\mathcal{E}_{\text{actual}}$. To our best knowledge, this is a novel computational problem, and has not been explored in the literature. While we are mainly concerned with sparse-spectrum signals as defined earlier, the model of *band-limited* signals – a subset of sparse-spectrum signals – has been widely considered in the literature (albeit not in conjunction with incorrectly defined graphs) Zhu & Rabbat (2012); Ortega et al. (2018). In the band-limited model, the signals are linear combinations of the first s eigenvectors with the smallest frequencies, with s being the band limit. In the following section, we present a literature review of closely related work.

2.1 Related Work

Graph Spectral Compressed Sensing: Recovery of graph signals from compressive measurements while making use of the graph structure via their GFT is sometimes called graph spectral compressed sensing Zhu & Rabbat (2012); Jung et al. (2018). While these and other works in GSP (e.g. see references in Ortega et al. (2018)) focus on band-limited signals, we mainly focus on the recovery of sparse-spectrum signals, i.e. signals with arbitrary sparse support in the GFT domain. In Zhu & Rabbat (2012); Jung et al. (2018); Zou et al. (2020), the measurements are the values of the graph signals on a subset of the nodes, and the purpose of compressive recovery is to interpolate the missing data at other nodes. Our work focuses on compressive measurements which are random linear combinations of the values at the different nodes, more in line with traditional CS or group testing Goenka et al. (2021), although we also consider graph signal interpolation. More importantly, we consider *perturbed* graphs.

Signals over Graphs with Partly Erroneous Topology: Our work is different as compared to a large body of work in graph learning (*eg*, Segarra et al. (2017) or the references in Dong et al. (2019)), which typically assumes *no* knowledge of the graph, but an availability of many graph signals defined over the nodes. Instead, we assume that the graph is known up to a few edge perturbations, and that only a single graph signal is indirectly observed via compressive measurements. The approach of correcting a small number of edges has been followed in a small number of papers, albeit in the non-compressive regime given a single signal vector – Ceci & Barbarossa (2020); Ceci et al. (2020). The work in Ceci & Barbarossa (2020) gives an approximate formula to compute eigenvectors of a perturbed graph from those of the original graph, and uses it in a brute-force algorithm which performs corrections to a graph topology given a band-limited graph signal with known band limit. A total least squares approach to robust graph signal recovery given structural equation models is presented in Ceci et al. (2020). The work in Miettinen et al. (2021) develops models for perturbations to edges in a graph and examines their effect on graph signal filtering and independent components analysis. As opposed to this, in our work, we present a method for robust graph signal recovery from *compressive measurements* given a small number of perturbations in the graph topology, and assuming the signal has a *sparse*, but not necessarily band-limited, representation in the underlying GFT basis. The work in Mahmood et al. (2018) performs joint recovery of an undirected, weighted, data-dependent graph with nodes representing overlapping patches of a 2-D image along with the image itself (i.e.,the graph signal) from compressive measurements. This is done by alternately re-computing the graph from the image, and then invoking graph total variation (GTV) and wavelet-sparsity prior for tomographic reconstruction. We note that their method is not applicable to our setting, where the graph is unweighted, and not data-dependent - i.e., it cannot be computed directly from the graph signal. Also, our technique focuses on recovery of a small number graph perturbations and not the full graph, and is not restricted to just graphs associated with images. In our setting, the signal is a linear combination of a small number of arbitrary GFT basis vectors, and hence the signals need not be smooth over the graph, whereas in

Mahmood et al. (2018), the GTV prior assumes piece-wise smoothness of the graph signal. Unlike Mahmood et al. (2018), we also provide sufficient conditions for successful signal and graph recovery using one of our methods in Sec. 3.3.

Compressed Sensing with Perturbed Models: We now give an overview of techniques in general CS that deal with model mismatches. Referring to Eqn. 1, we could have perturbations in either the sensing matrix $\boldsymbol{\Phi}$ or the representation matrix $\boldsymbol{\Psi}$ or both. The former problem has been explored in terms of perturbations to specified Fourier frequencies in Fourier sensing matrices in magnetic resonance imaging in work such as Pandotra et al. (2019); Ianni & Grissom (2016). More unstructured and dense perturbations to $\boldsymbol{\Phi}$ are considered in Zhu et al. (2011); Parker et al. (2011); Fosson et al. (2020). The focus of the work in this paper, however, is related to perturbations in $\boldsymbol{\Psi}$, because perturbations to the graph topology induces changes in the GFT matrix, which is the signal *representation* matrix (and not the sensing matrix which is completely independent). The problem of perturbations in $\boldsymbol{\Psi}$ has been extensively explored in the context of off-the-grid signal representation in sinusoidal bases such as the Fourier or discrete cosine transform in Chi et al. (2011); Nichols et al. (2014); Tan et al. (2014), using approaches such as alternating minimization Nichols et al. (2014), modified greedy algorithms like orthogonal matching pursuit (OMP) Teke et al. (2013), or structured sparsity Zhu et al. (2011); Tan et al. (2014). Applications to problems such as direction of arrival estimation have been explored. In contrast to this, we explore perturbations in the representation matrix in an *indirect* manner. That is, we explore methods to correct for perturbations in *edge specifications* in the adjacency matrix within our optimization framework. Given changes to the adjacency matrix, the Laplacian matrix and its eigenvectors are recomputed. We also note that small perturbations in the Laplacian matrix may lead to large perturbations of some of its eigenvectors – especially in those with high frequency. Hence the methods which make the assumption of small perturbation in $\boldsymbol{\Psi}$ Zhu et al. (2011) are not directly applicable to the problem considered in our work.

3 Method

3.1 Recovery using standard CS decoding algorithms

It is well known that a signal $\boldsymbol{x}^*$ acquired compressively as in Eqn. 1 and sparse in some known orthonormal basis $\boldsymbol{\Psi}$ may be recovered via Lasso (Least Absolute Shrinkage and Selection Operator) Hastie et al. (2015) as:

$$\text{Lasso:} \qquad \hat{\boldsymbol{x}}_{\text{lasso}} = \arg\min_{\boldsymbol{x}} \|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{x}\|_2^2 + \mu\|\boldsymbol{\Psi}^T\boldsymbol{x}\|_1, \tag{2}$$

where $\mu > 0$ is a regularization parameter. Suitable choices of matrix $\boldsymbol{\Phi}$, such as those whose entries are drawn independently from a zero-mean Gaussian, satisfy with high probability the Restricted Eigenvalue Condition (Hastie et al., 2015, Sec. 11.2.2) if $m = O(s \log n)$ or other similar properties sufficient for recovery of $\boldsymbol{x}^*$ using Lasso (Hastie et al., 2015, Thm. 11.1) or other CS decoding algorithms. The regularization parameter μ is typically chosen via cross-validation (CV) Zhang et al. (2014), by holding out some $m_{\text{cv}} < m$ measurements, performing recovery using only the remaining $m_{\text{r}} = m - m_{\text{cv}}$ measurements, and choosing the value of μ for which the CV error $\epsilon_{\text{cv}}^{\mu}$ (defined below) is minimized. As shown in (Zhang et al., 2014, Thm. 1), the CV error acts as a data-driven proxy for the (unobservable) mean-squared error. This estimator is represented as shown below:

$$\begin{aligned}
&\texttt{CVE}(\boldsymbol{y}, \boldsymbol{\Phi}, \boldsymbol{\Psi}, m_{\text{cv}}, \mu) : \hat{\boldsymbol{x}}_{\text{r}}^{\mu} = \operatorname*{argmin}_{\boldsymbol{x}} \|\boldsymbol{y}_{\text{r}} - \boldsymbol{\Phi}_{\text{r}}\boldsymbol{x}\|_2^2 + \mu\|\boldsymbol{\Psi}^T\boldsymbol{x}\|_1, \\
&\qquad \epsilon_{\text{cv}}^{\mu} = \|y_{\text{cv}} - \boldsymbol{\Phi}_{\text{cv}}\hat{\boldsymbol{x}}_{\text{r}}^{\mu}\|_2^2, \qquad (3) \\
&\text{Lasso-Cv}(\boldsymbol{y}, \boldsymbol{\Phi}, \boldsymbol{\Psi}, \Gamma, m_{\text{cv}}) : \hat{\mu} = \arg\min_{\mu \in \Gamma} \epsilon_{\text{cv}}^{\mu}, \\
&\qquad \hat{\boldsymbol{x}}_{\text{lasso-cv}} = \operatorname*{argmin}_{\boldsymbol{x}} \|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{x}\|_2^2 + \hat{\mu}\|\boldsymbol{\Psi}^T\boldsymbol{x}\|_1. \qquad (4)
\end{aligned}$$

Here `CVE` is the CV error computation routine, and Lasso-Cv is Lasso with cross-validation. Also, $\boldsymbol{y}_{\text{cv}}$ and $\boldsymbol{\Phi}_{\text{cv}}$ are the held out measurements and the corresponding rows of $\boldsymbol{\Phi}$, and $\boldsymbol{y}_{\text{r}}$ and $\boldsymbol{\Phi}_{\text{r}}$ are the remaining measurements or rows used for signal recovery. Furthermore, $\hat{\boldsymbol{x}}_{\text{r}}^{\mu}$ is the signal recovered using the parameter

value μ, $\epsilon_{\text{cv}}^{\mu}$ is the cross-validation error of $\hat{\boldsymbol{x}}_{\text{r}}^{\mu}$, Γ is the set of possible values of μ which are tried, $\hat{\mu}$ is the value of $\mu \in \Gamma$ which gives the smallest CV error, and $\hat{\boldsymbol{x}}_{\text{lasso-cv}}$ is the final estimate of $\boldsymbol{x}^*$ output by Lasso-Cv. Note that the cross-validation approach does *not* require knowledge of the underlying signal, or knowledge of the underlying actual graph or its eigenvectors, in any manner.

Hence an estimate of $\boldsymbol{x}^*$ could be recovered via Lasso (Eqn. 2) or Lasso-Cv (Eqn. 4) by putting $\boldsymbol{\Psi} = \boldsymbol{V}_{\text{actual}}$, with recovery guarantees from compressed sensing theory. A naive method of estimating $\boldsymbol{x}^*$ when $\mathcal{E}_{\text{actual}}$ is not known is to use the GFT matrix of the nominal graph, $\boldsymbol{V}_{\text{nominal}}$ as the orthonormal basis $\boldsymbol{\Psi}$ in Lasso or Lasso-Cv from Eqn. 4. Thus we have the following estimates:

$$\hat{\boldsymbol{x}}_{\text{actual}} = \text{Lasso-Cv}(\boldsymbol{y}, \boldsymbol{\Phi}, \boldsymbol{V}_{\text{actual}}, \Gamma, m_{\text{cv}}), \tag{5}$$

$$\hat{\boldsymbol{x}}_{\text{nominal}} = \text{Lasso-Cv}(\boldsymbol{y}, \boldsymbol{\Phi}, \boldsymbol{V}_{\text{nominal}}, \Gamma, m_{\text{cv}}). \tag{6}$$

We refer to the technique of using the GFT basis in Lasso as Gft-Lasso. If the actual graph is used, we call it Agft-Lasso, and if the nominal graph is used, it is called Ngft-Lasso. If cross-validation is used to determine the value of the parameter μ, then these techniques are termed Gft-Lasso-Cv, Agft-Lasso-Cv, and Ngft-Lasso-Cv, respectively.

Since the set of edges $\mathcal{E}_{\text{nominal}}$ differs slightly from $\mathcal{E}_{\text{actual}}$, the GFT matrix of the nominal graph, $\boldsymbol{V}_{\text{nominal}}$ will be a perturbed version of the actual GFT matrix $\boldsymbol{V}_{\text{actual}}$. Indeed, $\boldsymbol{x^*}$ may not even be sparse in $\boldsymbol{V}_{\text{nominal}}$ if the perturbation is significant. In the following subsections, we present methods which use the CV error of Gft-Lasso-Cv to select from potential refinements of the nominal graph. Note that even small perturbations to the adjacency matrix can lead to large differences between $\boldsymbol{V}_{\text{actual}}$ and $\boldsymbol{V}_{\text{nominal}}$, and so we cannot exploit any sparsity property of $\boldsymbol{V}_{\text{actual}} - \boldsymbol{V}_{\text{nominal}}$ for deriving $\boldsymbol{V}_{\text{actual}}$.

3.2 Greedy Edge Selection

We present the Greedy Edge Selection (Ges) algorithm (Alg. 1) to solve the CPGR problem. The main idea is to keep refining the edges of a candidate graph – initialized with the edges of the nominal graph – by perturbing one edge at a time, as long as the cross-validation error of the retrieved signal on a held-out set of measurements keeps decreasing. The hope is that each greedy refinement of the candidate graph brings it closer to the actual graph, such that eventually the actual graph as well as the original graph signal are recovered. The algorithm performs at most d_0 greedy refinement steps where d_0 is an upper bound on the number of edge perturbations. The edges of the candidate graph after greedy refinement step t are referred to as $\mathcal{E}_{\text{candidate}}^{(t)}$, with $\mathcal{E}_{\text{candidate}}^{(0)} = \mathcal{E}_{\text{nominal}}$. In the greedy step t of the algorithm, all graphs which can be obtained by adding or removing one edge to or from the edge set of the candidate graph $\mathcal{E}_{\text{candidate}}^{(t-1)}$ (except those which have already been perturbed in greedy steps $1, \ldots, (t-1)$) are considered. There are $\binom{n}{2} - (t-1)$ such graphs. GFT matrices of each of these graphs are computed via eigendecomposition of their Laplacian matrices. Gft-Lasso-Cv (Sec. 3.1) is performed to find the smallest CV error and the ideal value of μ using each GFT matrix. The edge $e_{\text{best}}^{(t)}$ which gives the smallest value for CV error is chosen and the candidate graph is updated, provided the CV error decreases by more than a factor $\tau \in (0, 1]$ relative to the CV error for the current candidate graph. Otherwise, the algorithm stops, and an estimate of the signal is returned by performing Lasso using the GFT matrix and μ obtained from $\mathcal{E}_{\text{candidate}}^{(t-1)}$.

Noise-based stopping criterion: From Eqn. 3 and Eqn. 1, we see that the CV error for the ground truth signal $\boldsymbol{x}^*$ is a sum of squares of m_{cv} i.i.d. Gaussian random variables with mean 0 and variance σ^2, and thus has mean equal to $m_{\text{cv}}\sigma^2$ and standard deviation $\sqrt{2m_{\text{cv}}}\sigma^2$. Hence, before a greedy step begins, the Ges algorithm checks whether the CV error is within a high confidence interval of $m_{\text{cv}}\sigma^2$ and stops if that is the case, in order to prevent fitting to the measurement noise.

Brute-force algorithm: A brute-force version of Ges (referred to as Brute-Force Graph Selection or Bfgs) is also possible, wherein all graphs which are at most d_0 perturbations from $\mathcal{G}_{\text{nominal}}$ are considered, signals are recovered using their GFT matrices on m_{r} measurements, and the graph which gives the minimum CV error on the remaining m_{cv} measurements is chosen for final signal recovery. As there are $\binom{n}{2}$ possible edges, the total number of graphs enumerated via brute-force is $\binom{\binom{n}{2}}{0} + \binom{\binom{n}{2}}{1} + \cdots + \binom{\binom{n}{2}}{d_0} = O(n^{2d_0})$.

Algorithm 1 Greedy Edge Selection (GES)

Input: $\boldsymbol{y}$: Compressive measurements, $\boldsymbol{\Phi}$: measurement matrix, $\mathcal{E}_{\text{nominal}}$: nominal graph edge set, σ^2 : variance of measurement noise, Γ : values of μ for grid search, $\tau \in (0, 1]$: CV error improvement factor, m_{cv} : Number of CV measurements, d_0 : maximum number of edge perturbations, g : CV error confidence interval factor

Output: Estimated graph signal $\hat{\boldsymbol{x}}_{\text{greedy}}$

Initialize: $\mathcal{E}^{(0)}_{\text{candidate}} \leftarrow \mathcal{E}_{\text{nominal}}$, $\boldsymbol{V}^{(0)} \leftarrow \boldsymbol{V}_{\text{nominal}}$, $\{\epsilon^{(0)}_{\text{cv}}, \mu^{(0)}\} \leftarrow \{\min, \underset{\mu\in\Gamma}{\text{argmin}}\}\texttt{CVE}(\boldsymbol{V}^{(0)}, \mu|\boldsymbol{y}, \boldsymbol{\Phi}, m_{\text{cv}})$, $\mathcal{P}^{(0)} = \emptyset$, and $t_{\text{est}} \leftarrow 0$
for t in $1, \ldots, d_0$ **do**
  if $\epsilon^{(t-1)}_{\text{cv}} \leq (m_{\text{cv}}\sigma^2 + g\sqrt{2m_{\text{cv}}}\sigma^2)$ **then**
    break, to prevent fitting on noise
  for each possible edge e which is not in $\mathcal{P}^{(t-1)}$ **do**
    Obtain perturbed graph edge set: $\mathcal{E}^{(t)}_e \leftarrow (\mathbb{E}^{(t-1)}_{\text{candidate}} - \{e\}) \cup (\{e\} - \mathbb{E}^{(t-1)}_{\text{candidate}})$
    Compute Laplacian matrix $\boldsymbol{L}^{(t)}_e$ from $\mathcal{E}^{(t)}_e$ and GFT matrix $\boldsymbol{V}^{(t)}_e$ via eigendecomposition of $\boldsymbol{L}^{(t)}_e$
    Compute best CV error and μ using $\boldsymbol{V}^{(t)}_e$: $\epsilon^{(e,t)}_{\text{cv}} \leftarrow \min_{\mu\in\Gamma} \texttt{CVE}(\boldsymbol{V}^{(t)}_e, \mu|\boldsymbol{y}, \boldsymbol{\Phi}, m_{\text{cv}})$, $\mu^{(t)}_e \leftarrow \underset{\mu\in\Gamma}{\text{argmin}}\, \texttt{CVE}(\boldsymbol{V}^{(t)}_e, \mu|\boldsymbol{y}, \boldsymbol{\Phi}, m_{\text{cv}})$
  if $\min_e \epsilon^{(e,t)}_{\text{cv}} < \tau\epsilon^{(t-1)}_{\text{cv}}$ **then**
    Select edge: $e^{(t)}_{\text{best}} \leftarrow \arg\min_e \epsilon^{(e,t)}_{\text{cv}}$
    Perform updates: $\mathcal{P}^{(t)} \leftarrow P^{(t-1)} \cup \{e^{(t)}_{\text{best}}\}$, $\epsilon^{(t)}_{\text{cv}} \leftarrow \min_e \epsilon^{(e,t)}_{\text{cv}}$, $\mathbb{E}^{(t)}_{\text{candidate}} \leftarrow \mathbb{E}^{(t)}_{e^{(t)}_{\text{best}}}$, $\boldsymbol{V}^{(t)} \leftarrow \boldsymbol{V}^{(t)}_{e^{(t)}_{\text{best}}}$, $\mu^{(t)} \leftarrow \mu^{(t)}_{e^{(t)}_{\text{best}}}$, and $t_{\text{est}} \leftarrow t$
  else
    break; insignificant improvement.
Estimate $\hat{\boldsymbol{x}}_{\text{greedy}}$ via LASSO from $\boldsymbol{y}$, $\boldsymbol{\Phi}$, $\boldsymbol{V}^{(t_{\text{est}})}$, and $\mu^{(t_{\text{est}})}$ using all the measurements: $\hat{\boldsymbol{x}}_{\text{greedy}} \leftarrow \underset{\boldsymbol{x}\in R^n}{\arg\min} \|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{x}\|_2^2 + \mu^{(t_{\text{est}})}\|\boldsymbol{V}^{(t_{\text{est}})T}\boldsymbol{x}\|_1$
return $\hat{\boldsymbol{x}}_{\text{greedy}}$

Running time: The algorithm performs a maximum of d_0 greedy steps. In each greedy step, GFT-LASSO is performed for a maximum of $\binom{n}{2}$ graphs. Grid search for μ is performed over $|\Gamma|$ values. Hence the total number of GFT-LASSO optimizations performed is $O(|\Gamma|d_0n^2)$, which is a significant improvement over the $O(|\Gamma|n^{2d_0})$ GFT-LASSO optimizations performed by BFGS.

Signal recovery accuracy: While the greedy algorithm is not guaranteed to recover the original graph and the signal, we find empirically (Sec. 5.1) that the mean error in the recovered signal is much lower than that using the nominal graph, and in many cases, the original graph is recovered. In Theorem 3.3, we present conditions under which the solution is guaranteed to improve at any step of the GES algorithm.

3.3 Recovery Guarantees and Bounds

Let $\hat{\boldsymbol{x}}_{\mathcal{G}}$ denote the signal recovered using GFT-LASSO-CV with the GFT of graph $\mathcal{G}$. We make the assumption that each entry of $\boldsymbol{\Phi}$ is independently drawn from a sub-Gaussian distribution of mean 0 and variance 1. We present the following recovery guarantee for the BFGS algorithm from Sec. 3.2: [Brute-Force Algorithm Recovery Guarantee] If in the BFGS algorithm in Sec. 3.2, the number of CV measurements m_{cv} obeys

$$m_{\text{cv}} \;\geq\; 4\Big(1 + \frac{2c}{(c-1)^2}\Big)\Big\{\ln|\Gamma| + \ln(d_0+1) + 2d_0\ln n + \ln\frac{1}{\delta}\Big\} \tag{7}$$

for arbitrary constants $c \in (1, \infty)$ and $\delta \in (0, 1)$, then its recovery error is bounded as

$$\|\hat{\boldsymbol{x}}_{\text{bf}} - \boldsymbol{x}^*\|_2^2 \ < \ c\|\hat{\boldsymbol{x}}_{\text{actual}} - \boldsymbol{x}^*\|_2^2 + (c-1)\sigma^2 \tag{8}$$

with probability more than $1 - \delta$. As a consequence, if

$$\|\hat{\boldsymbol{x}}_{\mathcal{G}} - \boldsymbol{x}^*\|_2^2 \ \geq \ c\|\hat{\boldsymbol{x}}_{\text{actual}} - \boldsymbol{x}^*\|_2^2 + (c-1)\sigma^2, \tag{9}$$

for all graphs $\mathcal{G} \neq \mathcal{G}_{\text{actual}}$ which are upto d_0 edge perturbations from $\mathcal{G}_{\text{nominal}}$, then with probability more than $1 - \delta$, $\hat{\boldsymbol{x}}_{\text{bf}} = \hat{\boldsymbol{x}}_{\text{actual}}$, and the actual graph is recovered. The proof is provided in the supplemental material (Sec. **??**). It is based on a theorem from Zhang et al. (2014) regarding how well the CV error predicts the recovery error for compressed sensing with Gaussian random matrices[1]. We use this theorem and the union bound to lower bound the probability that the CV error of the signal recovered using GFT-LASSO-CV with the actual graph is lower than the CV errors for all signals having a recovery error satisfying the condition in Eqn. 9.

Discussion on Theorem 3.3:

1. Given compressive measurements for signals defined on an incorrectly defined graph, it is not *a priori* clear whether the problem of signal recovery and graph recovery are solvable at all. The main contribution of Theorem 3.3 is to outline conditions where successful recovery of the signal is possible (Eqns. 7 and 8), and where successful recovery of the graph is possible (Eqn. 9). The result shows that the number of additional measurements needed for cross-validation depends scales as $O(d_0 \log n)$. Therefore $m_{\text{cv}} \ll n$ for appropriate choices of the other parameters, and signal and graph recovery from *compressive* measurements is possible under appropriate conditions without too many additional measurements.

2. There is a tradeoff between the number of CV measurements used and the quality of recovery, encapsulated by the parameter c. If c is close to 1, the recovery error is close to or equal to that achieved using GFT-LASSO-CV on the actual graph. However in such a case, m_{cv} will be large. On the other hand, a large value of c will allow for a smaller value of m_{cv} – however in that case, the recovered signal may have error higher than if the actual graph was known. In practice, our method attains significant reduction in signal error compared to the baseline method of using the GFT of the nominal graph (Sec. 5.1).

3. Dependence of the bound on d_0 in Eqn. 7 shows that the method works well if the nominal graph and the actual graph are only a few perturbations away from each other. In particular, if a nominal graph were not known at all, then the brute-force algorithm must enumerate all graphs, making $d_0 = \binom{n}{2}$. In this case, $m_{\text{cv}} = \Omega(n^2 \ln n)$, and we are no longer in the regime of CS.

4. The upper bound on $\|\hat{\boldsymbol{x}}_{\text{actual}} - \boldsymbol{x}^*\|_2^2$ scales as $O(s \log(n)/m)$ (Hastie et al., 2015, Thm. 11.1, Eqn. 11.15) where the sparsity level $s := \|\boldsymbol{V}_{\text{actual}}^T \boldsymbol{x}^*\|_0$. For successful compressive recovery, we typically require $O(s \log n)$ measurements. Thus as s increases, the upper bound on $\|\hat{\boldsymbol{x}}_{\text{actual}} - \boldsymbol{x}^*\|_2^2$ and the number of measurements needed for successful recovery, both increase. As $\|\hat{\boldsymbol{x}}_{\text{bf}} - \boldsymbol{x}^*\|_2^2$ is upper bounded by $\|\hat{\boldsymbol{x}}_{\text{actual}} - \boldsymbol{x}^*\|_2^2$, the former is also indirectly affected by s.

5. The dependence of the bound in Eqn. 7 on $|\Gamma|$ seems to be an artifact of our proof technique. Our proof does not exploit the fact that there may be many 'bad' graphs in the search space, for which the recovery error (and the CV error) will be high regardless of the value of $\mu \in \Gamma$ used. Instead, the proof proceeds by assuming that the CV errors for a 'bad' graph with different values of μ are not correlated, leading to an overestimation of the bound for m_{cv}. Perhaps some other proof technique might be used which removes or weakens the dependence of the bound on $|\Gamma|$.

[1]The theorems in Zhang et al. (2014) can however be easily extended to handle any sub-Gaussian matrix by using an appropriate formula for the variance of the particular sub-Gaussian distribution used in Eqn. 50 in Zhang et al. (2014), which will reflect accordingly in the bounds in Theorem 3.3.

6. **Remark on Graph Recovery:** While the brute-force method is guaranteed to recover the signal to good accuracy as per Theorem 3.3, we do not know if it always recovers the actual graph, even in the absence of measurement noise. For example, there might be cases wherein a graph signal has a sparser or equally-sparse representation in the GFT basis of a graph which is not the actual graph, which might get chosen by the algorithm. In particular, it is known that the eigenvectors of the Laplacian matrix of a graph and its complement graph[2] are the same. In general, an edge perturbs an eigenvector only if the entries of the eigenvector on the two nodes of the edge are not equal (see Eqn. 13). The first eigenvector of the graph Laplacian never gets perturbed, since it has the same value on all nodes. Another example is the case of a graph with more than one connected component, in which the eigenvectors have non-zero entries in only one component each. Perturbation of an edge in one component will not affect the eigenvectors which have only zero entries in nodes belonging to that component. Thus this perturbed edge will not be recoverable if the graph signal was a linear combination of eigenvectors which only had zero entries in the corresponding component. The authors do not know if there exist connected graphs which are a small number of perturbations from each other and share some eigenvectors or have eigenvectors which are the linear combination of a small number of the eigenvectors of the other graph. Consider another example consisting of two complete graphs which are joined to each other via a few connecting edges to create a larger combined graph. Consider a signal created by a linear combination of a few low frequency GFT basis vectors of the larger graph. The deletion of an edge belonging to one of the smaller complete graphs will not have much of an effect on the GFT of the perturbed graph. For such perturbations, the sufficient condition in Eqn. 9 will be violated (because the error $\|\boldsymbol{x}^* - \hat{\boldsymbol{x}}_{\mathcal{G}}\|_2$ for many candidate graphs $\mathcal{G}$ will be very close to $\|\boldsymbol{x}^* - \hat{\boldsymbol{x}}_{\text{actual}}\|_2$), even more so under high noise, and hence the actual graph will not be recoverable. In this way, the structure of $\mathcal{G}_{\text{actual}}$ affects graph recovery even though it may not affect signal recovery. Interestingly, the signal recovery bound in Eqn. 8 does not depend on the structure of the graph – it only depends on m_{cv}, σ and (indirectly) on s, m_r.

7. We have also derived a result analogous to Theorem 1 but using CV errors instead of MSEs. This result produces a *testable* condition for signal and graph recovery. It is presented as Theorem S.5 of the supplemental.

Analogous to Theorem 3.3, we present a theorem for solution improvement using the greedy GES method (Algorithm 1). Let $\hat{\boldsymbol{x}}^{(t)}$ be the estimate by the greedy edge selection algorithm after t steps, and let $\hat{\boldsymbol{x}}^{(t)}_{\text{best}}$ be the estimate with the lowest recovery error amongst the signals recovered at step t.

[Greedy Edge Selection Solution Improvement Guarantee] If in the greedy edge selection algorithm (Algorithm 1) with $\tau = 1$ the number of CV measurements m_{cv} obeys

$$m_{\text{cv}} \;\geq\; 4\Big(1 + \frac{2c}{(c-1)^2}\Big)\Big\{\ln|\Gamma| + \ln\frac{n(n+1)}{2} + \ln\frac{1}{\delta}\Big\} \tag{10}$$

for arbitrary constants $c \in (1, \infty)$ and $\delta \in (0, 1)$, then the recovery error after step t is bounded as

$$\|\hat{\boldsymbol{x}}^{(t)} - \boldsymbol{x}^*\|_2^2 \;<\; c\|\hat{\boldsymbol{x}}^{(t)}_{\text{best}} - \boldsymbol{x}^*\|_2^2 + (c-1)\sigma^2 \tag{11}$$

with probability more than $1 - \delta$. As a consequence, if

$$\|\hat{\boldsymbol{x}}^{(t-1)} - \boldsymbol{x}^*\|_2^2 \;\geq\; c\|\hat{\boldsymbol{x}}^{(t)}_{\text{best}} - \boldsymbol{x}^*\|_2^2 + (c-1)\sigma^2, \tag{12}$$

then the solution is guaranteed to improve at step t with probability more than $1 - \delta$. The proof for Theorem 3.3 is similar to that for Theorem 3.3, and is provided in the supplemental material. The main difference is in the lower bound for m_{cv}, which is due to the different number of perturbed graphs considered in each algorithm. Theorem 3.3 outlines conditions under which the solution at the end of each iteration of GES has a sensibly bounded error, and under which the solution improves from iteration to iteration (where an iteration consists of perturbing one extra edge over and above the previous iteration). The condition in

[2]The complement of a graph is a graph in which every edge present in the original graph is absent in the complement graph and every edge which is absent in the original graph is present in the complement graph.

the theorem states a requirement of only a *logarithmic* number of additional compressive measurements for CV.

Case of Non-distinct eigenvalues: We have so far assumed that the eigenvalues of the Laplacian of the nominal and actual graphs are distinct. In cases where eigenvalues of the Laplacian are not distinct, there will be no adverse effect on signal reconstruction if the (randomly generated and sparse) signal support does not cover eigenvectors associated with eigenvalues having multiplicity greater than one. In case the signal support does include eigenvectors associated with eigenvalues having multiplicity greater than one, the compressive reconstruction will still not be affected if the multiplicity is not too much larger than 1. This is because the effective increase in signal ℓ_0 norm will be upper bounded by the total eigenvalue multiplicity.

3.4 Alternatives to Eigendecomposition for GFT basis computation

The GES algorithm requires computing the eigen-decomposition (step 16 in Alg. 1) of the Laplacian matrix $\boldsymbol{L}_{\boldsymbol{e}}^{(t)}$ of the graph obtained by perturbing the candidate edge set $\mathcal{E}_{\text{candidate}}^{(t-1)}$ with edge e, for each possible edge e, at each time step t. This step may take a long time for large graphs. It may be made more efficient by using an approximate formula given in Ceci & Barbarossa (2020), to obtain the eigenvectors of $\boldsymbol{L}_{\boldsymbol{e}}^{(t)}$ from the eigenvectors of $\boldsymbol{L}_{\text{nominal}}$, and the set of perturbations $\mathcal{P}^{(t-1)} \cup \{e\}$, instead of performing eigendecomposition of $\boldsymbol{L}_{\boldsymbol{e}}^{(t)}$. If an original Laplacian matrix $\boldsymbol{L}$ is perturbed by a set of edges $\mathcal{P}$ to obtain a perturbed Laplacian matrix $\tilde{\boldsymbol{L}}$, and their eigenvectors are $\boldsymbol{v}_1 \dots \boldsymbol{v}_n$ and $\tilde{\boldsymbol{v}}_1 \dots \tilde{\boldsymbol{v}}_n$ respectively, then from (Ceci & Barbarossa, 2020, Eqn. 7 and 11) we have the approximation:

$$\tilde{\boldsymbol{v}}_k \simeq \boldsymbol{v}_k + \sum_{\{i,j\}\in\mathcal{P}} \sigma_{i,j}(\boldsymbol{v}_k(i) - \boldsymbol{v}_k(j)) \sum_{\substack{l=2 \\ l\neq k}}^{n} \frac{\boldsymbol{v}_l(i) - \boldsymbol{v}_l(j)}{\lambda_k - \lambda_l} \boldsymbol{v}_l, \tag{13}$$

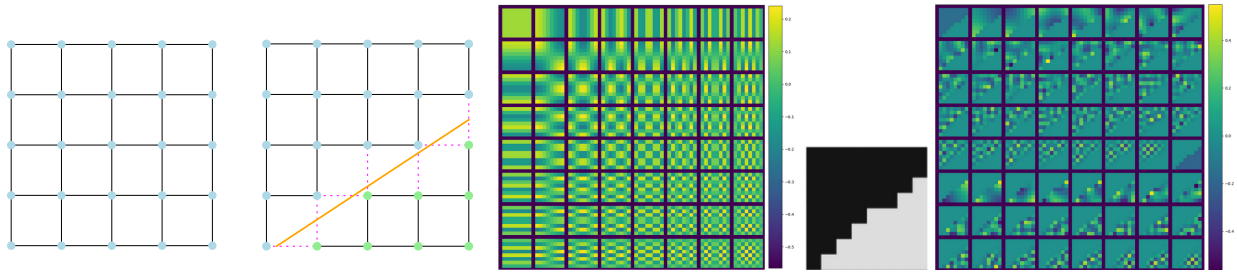
where $\sigma_{i,j} = 1$ for edge addition, and $\sigma_{i,j} = -1$ for edge deletion. The approximation for $\tilde{\boldsymbol{v}}_k$ is valid only under the condition that

$$\lambda_{k-1} - \lambda_k \ll \sum_{\{i,j\}\in\mathcal{P}} \sigma_{i,j}(\boldsymbol{v}_k(i) - \boldsymbol{v}_k(j))^2 \ll \lambda_{k+1} - \lambda_k. \tag{14}$$

We discuss some intuition behind this approximation from Ceci & Barbarossa (2020) and the condition under which it is valid. First, note that each new edge $\{i,j\}$ in $\mathcal{P}$ introduces a rank-one perturbation of $\boldsymbol{L}$ with the matrix $\sigma_{i,j}\boldsymbol{a}_{i,j}\boldsymbol{a}_{i,j}^T$, where $\boldsymbol{a}_{i,j}$ is a vector with $\boldsymbol{a}_{i,j}(i) = 1, \boldsymbol{a}_{i,j}(j) = -1$, and the remaining entries of $\boldsymbol{a}_{i,j}$ are equal to zero. That is, $\tilde{\boldsymbol{L}} = \boldsymbol{L} + \boldsymbol{\Delta L}$, where the perturbing matrix $\boldsymbol{\Delta L} = \sum_{\{i,j\}\in\mathcal{P}} \sigma_{i,j}\boldsymbol{a}_{i,j}\boldsymbol{a}_{i,j}^T$. Note that $\boldsymbol{a}_{i,j}^T\boldsymbol{v}_k = (\boldsymbol{v}_k(i) - \boldsymbol{v}_k(j))$ and hence $\boldsymbol{v}_k^T\boldsymbol{\Delta L}\boldsymbol{v}_k = \sum_{\{i,j\}\in\mathcal{P}} \sigma_{i,j}\boldsymbol{v}_k^T\boldsymbol{a}_{i,j}\boldsymbol{a}_{i,j}^T\boldsymbol{v}_k = \sum_{\{i,j\}\in\mathcal{P}} \sigma_{i,j}(\boldsymbol{v}_k(i) - \boldsymbol{v}_k(j))^2$. It is easily verified that if $\boldsymbol{a}_{i,j}^T\boldsymbol{v}_k = 0$ (i.e. $(\boldsymbol{v}_k(i) - \boldsymbol{v}_k(j)) = 0$) for all the perturbing edges $\{i,j\} \in \mathcal{P}$, then $\boldsymbol{v}_k$ is also an eigenvector of $\tilde{\boldsymbol{L}}$. Recall that the normalized eigenvectors of a matrix $\boldsymbol{L}$ are the critical points of the Rayleigh quotient $\boldsymbol{x}^T\boldsymbol{L}\boldsymbol{x}$ on the unit sphere $\boldsymbol{x}^T\boldsymbol{x} = 1$, with the eigenvalue being the value of the Rayleigh quotient at the corresponding eigenvector. Eqn. 14 means that the perturbation of the Rayleigh quotient at $\boldsymbol{v}_k$ must be much smaller than the difference in values of the Rayleigh quotient at the closest critical points. In Eqn. 13, the perturbation of $\boldsymbol{v}_k$ by edge $\{i,j\}$ is negligible if $|\boldsymbol{a}_{i,j}^T\boldsymbol{v}_k|$ is small. Similarly, if $|\boldsymbol{a}_{i,j}^T\boldsymbol{v}_l|$ is small, then the perturbation of other eigenvectors in the direction of $\boldsymbol{v}_l$ due to the edge $\{i,j\}$ is small. Finally, a small eigen-gap $\lambda_k - \lambda_{k-1}$ or $\lambda_{k+1} - \lambda_k$ means that the Rayleigh quotient is relatively flat between the two critical points, and hence the critical point may be changed easily via a perturbation. We evaluate the suitability of this approximation to our method in Sec. 5.1.

4 Application: Inferred Linear-Edge Compressive Image Recovery

We now present an application of the Compressive Perturbed Graph Recovery framework, in the domain of compressive image recovery. In this section, we use the term 'edge' as an element of a graph connecting a pair of vertices, and use the term 'image edge' to refer to the entity which separates two regions in an image.



(a) **(b)** **(c)** **(d)** **(e)**

Figure 1: (a) A 5×5 2-D lattice graph whose nodes represent pixels of a 5×5 patch and whose edges represent connections between a pixels and its four neighbors. This forms the nominal graph for the problem of compressive image patch recovery. (b) The lattice graph partitioned by an image edge (orange line). The graph edges going across the image edge are removed (dotted purple lines). Since the image edge is unknown before reconstruction, the image-edge partitioned graph is the (unknown) actual graph for the problem of recovery of an image patch from compressive measurements. (c) All 64 2-D DCT basis vectors of an 8×8 patch. (d) An 8×8 patch with a sharp edge. (e) Segmentation-aware basis vectors for this patch, obtained by computing the eigenvectors of the Laplacian matrix of the graph created by dropping the edges of the 8×8 lattice graph whose endpoints lie in different segments of the patch.

The GES method presented in Sec. 3.2 is a generally applicable technique on any graph. However, if the graph has some well-known structure to it, it may be possible to select the edges for perturbation in a more structured manner, instead of greedily one at a time. We present such a method for image reconstruction from compressive measurements taken using a block-based version Kulkarni et al. (2016) of the Rice Single-Pixel Camera Duarte et al. (2008). That is, for each $h \times w$ patch $\boldsymbol{x}^*$ of an image, compressive measurements $\boldsymbol{y}$ obtained via a noisy version of $\boldsymbol{\Phi x}^*$ as in Eqn. 1, are assumed to be available, with $\boldsymbol{\Phi}$ having dimensions $m \times n$ with $n = hw$ and $m \ll n$, and the task is to reconstruct each patch of the image.

Any 2-D (two-dimensional) grayscale raster image can be considered to be a graph signal, with each pixel being a node and horizontal and vertical neighbours connected to each other via an edge, with each pixel node mapped to its value in the image. Such a graph is called a 'lattice graph' (Fig. 1(a)). The 2-D Discrete Cosine Transform (DCT) basis (Fig. 1(c)) is known to form an eigenbasis of the Laplacian matrix of a 2-D lattice graph (Fracastoro et al., 2016, Proposition 1), and thus is a GFT basis for this graph. Piece-wise smooth images, depth-maps, and natural images have a sparse or compressible representation in the DCT basis, because the low-frequency basis vectors in DCT model the correlation between neighbouring pixels well. This fact is heavily exploited in the well known JPEG standard for image compression. Thus, it is natural to employ LASSO (Eqn. 2) with the DCT basis in order to reconstruct the image patches.

However, neighbouring pixels on either side of a sharp image-edge will not have correlated values, due to which DCT-based recovery will not be accurate near an image-edge. In such cases, it may be better to use some other transform basis whose vectors maintain no correlation in their values across the image edge. Since the DCT basis is a GFT basis of the 2-D lattice graph, one possible basis is the GFT basis of the graph obtained by dropping those edges of the 2-D lattice graph which link nodes that are located on two different sides of an image-edge – see Fig. 1(d) and Fig. 1(e) for an example of such a 'segmentation-aware' basis. We may call this graph the 'image-edge partitioned graph' (see Fig. 1(b)) if it is constructed using the true image edges. However, since the image is acquired compressively, it is not straightforward to know the exact location of the image-edges, and hence the image-edge partitioned graph is unknown. Hence, the problem of decoding of the compressively acquired image using the GFT of the unknown image-edge-partitioned graph while knowing only the 2D lattice graph may be formulated as CPGR, i.e. Problem 2. The 2D lattice graph is the nominal graph, the image-edge partitioned graph is the actual graph, and the set of pixel values is the unknown graph signal which is sparse/compressible in the GFT of the *actual* graph. In this application, we may consider the notion that the 'actual' graph that we seek to determine, is simply one that yields a sparser representation for the image patch, thereby allowing possibly better image reconstruction.

Algorithm 2 Inferred Linear-Edge Compressive Image Recovery (ILECIR)

Input: $\boldsymbol{y}$: vector of m measurements of an $h \times w$ patch, $\boldsymbol{\Phi}$: $m \times n$ measurement matrix with $n = hw$, $\boldsymbol{V}_{\text{DCT}}$: 2-D DCT basis vectors of an $h \times w$ patch, ϕ : set of linear image-edges connecting boundary pixels of an $h \times w$ patch, $\mathcal{V}_{h,w}$: GFT matrices of the partitioned graphs formed by the image-edges in ϕ, σ^2 : measurement noise variance, Γ : values of μ for grid search, m_{cv} : number of CV measurements, $\tau \in (0, 1]$: CV error improvement factor, g : CV error confidence interval factor.
Output: $\hat{P}$: estimated image patch of size $h \times w$

Compute CV error and μ using DCT:
$\{\epsilon_{\text{cv}}^{\text{DCT}}, \mu_{\text{DCT}}\} \leftarrow \{\min, \underset{\mu \in \Gamma}{\text{argmin}}\}\ \texttt{CVE}(\boldsymbol{V}_{\text{DCT}}, \mu | \boldsymbol{y}, \boldsymbol{\Phi}, m_{\text{cv}})$
if $\epsilon_{\text{cv}}^{\text{DCT}} \leq (m_{\text{cv}}\sigma^2 + g\sqrt{2m_{\text{cv}}}\sigma^2)$ i.e. close to noise **then**
  $\boldsymbol{V}_{\text{est}} \leftarrow \boldsymbol{V}_{\text{DCT}}$, $\mu_{\text{est}} \leftarrow \mu_{\text{DCT}}$
else
  for Each linear image edge $i \in \phi$ **do**
    Retrieve corresponding GFT matrix $\boldsymbol{V}_i \in \mathcal{V}_{h,w}$
    Compute best CV error and μ using i:
    $\{\epsilon_{\text{cv}}^{(i)}, \mu_i\} \leftarrow \{\min, \underset{\mu \in \Gamma}{\text{argmin}}\}\ \texttt{CVE}(\boldsymbol{V}_i, \mu | \boldsymbol{y}, \boldsymbol{\Phi}, m_{\text{cv}})$
  if $\min_i \epsilon_{\text{cv}}^{(i)} < \tau \epsilon_{\text{cv}}^{\text{DCT}}$ **then**
    $\boldsymbol{V}_{\text{est}} \leftarrow \boldsymbol{V}_i$, $\mu_{\text{est}} \leftarrow \mu_i$
Estimate the vectorized patch using LASSO: $\hat{\boldsymbol{x}}_{\text{ilecir}} \leftarrow \underset{x \in R^n}{\arg\min} \|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{x}\|_2^2 + \mu_{\text{est}} \|\boldsymbol{V}_{\text{est}}^T \boldsymbol{x}\|_1$
Convert $\hat{\boldsymbol{x}}_{\text{ilecir}}$ to $h \times w$ patch $\hat{P}$, assuming row-major order
return $\hat{P}$

Rather than directly applying the GES algorithm from Sec. 3.2 to solve this problem, it is better to exploit the inherent structure in image-edges to drop graph edges in a systematic way. With this in mind, we present a variant of GES called Inferred Linear-Edge Compressive Image Recovery (ILECIR) in Algorithm 2. Each image patch is assumed to either contain no image-edge or at most a single *linear* image-edge, i.e., a straight line with its endpoints at two boundary pixels of the patch. Reconstruction of the patch from a reconstruction subset of measurements is performed using the GFTs corresponding to each image-edge partitioned graph as well as DCT, and CV errors on a held-out subset of measurements are computed. The GFT corresponding to the estimated signal with the lowest cross-validation error is selected, and the final estimate of the patch is reconstructed using all the measurements. The noise-based stopping criterion from Sec. 3.2 is employed here.

We now present a theorem (proved in the supplemental material) showing how the ILECIR (Algorithm 2) solution improves upon the DCT-based recovery (DCT-LASSO-CV) solution. Let $\hat{\boldsymbol{x}}_{\text{best-edge}}$ denote the (vectorized) patch estimate with the lowest recovery error amongst all the patch estimates in the ILECIR algorithm for a single patch, and let $\hat{\boldsymbol{x}}_{\text{dct}}$ denote the estimate using DCT-based recovery. Note that $\hat{\boldsymbol{x}}_{\text{best-edge}}$ is unobservable. [ILECIR Solution Improvement Guarantee] If in the ILECIR algorithm (Algorithm 1) with $\tau = 1$ the number of CV measurements m_{cv} obeys

$$m_{\text{cv}} \;\geq\; 4\Big(1 + \frac{2c}{(c-1)^2}\Big)\Big\{\ln|\Gamma| + \ln(|\phi| + 1) + \ln\frac{1}{\delta}\Big\} \tag{15}$$

for arbitrary constants $c \in (1, \infty)$ and $\delta \in (0, 1)$ and where ϕ is the set of all possible linear image edges in an $h \times w$ patch with endpoints at the boundary, then the recovery error is bounded as

$$\|\hat{\boldsymbol{x}}_{\text{ilecir}} - \boldsymbol{x}^*\|_2^2 \;<\; c\|\hat{\boldsymbol{x}}_{\text{best-edge}} - \boldsymbol{x}^*\|_2^2 + (c-1)\sigma^2 \tag{16}$$

with probability more than $1 - \delta$. As a consequence, if

$$\|\hat{\boldsymbol{x}}_{\text{dct}} - \boldsymbol{x}^*\|_2^2 \;\geq\; c\|\hat{\boldsymbol{x}}_{\text{best-edge}} - \boldsymbol{x}^*\|_2^2 + (c-1)\sigma^2, \tag{17}$$

then the solution is guaranteed to improve over DCT-LASSO-CV with probability more than $1-\delta$. For the case of compressive image recovery, ILECIR is faster than GES (see Sec. S.V of the supplemental for more details) and also exploits the continuous/linear nature of the image edges unlike GES.

5 Empirical Evaluation

We perform an empirical evaluation of our GES and ILECIR methods, the details of which are discussed in Sec. 5.1 and 5.2 respectively.

5.1 Greedy Edge Selection on synthetic graphs

Experiment Setup: We test GES on the following graphs commonly used in the network science literature: Planted Partition Model (PPM), Stochastic Block Model (SBM), Erdős–Rényi (ER) Graph, Random Geometric Graph (RGG), Barabasi-Albert (BA) Graph and Karate Club (KC) graph. All these are random graph models, other than the KC graph, which is a two-community social network having $n = 34$ nodes. More details regarding the random graph models can be found in the supplemental material in Sec. **??**. The actual graphs used in our experiments are instantiated with $n = 100$ nodes for the random graph models. The PPMs have $l = 5$ communities, each of size $r = 20$, with $p = 0.9$ and $q = 0.01$ (intra-cluster and inter-cluster edge probabilities, respectively), with the expected number of edges being 895. The SBMs have $l = 5$ communities with sizes $\{r_1, \ldots, r_5\} = \{5, 10, 20, 25, 40\}$, and the probabilities p, q are set to be the same as the PPM. The ER Graphs have $p = \frac{895}{4950}$. The RGGs have $r = 0.27$. The BA graphs have $r = 10$. The graph parameters are chosen so that the expected number of edges of the graphs are roughly equal, except the SBM, for which the edge probabilities are matched with the PPM.

Nominal graphs are generated by perturbing $d \in \{1, 2, 5, 10\}$ edges of the actual graph. The set of edges to be perturbed – known as the perturbation set ω, with $|\omega| = d$ – is chosen to be a subset of a prior set Ω of 100 "potentially faulty edges" decided at the time of data generation. For each graph, all $\binom{n}{2}$ possible edges are assigned to different categories, and an approximately equal number of edges from each of these categories are sampled to form Ω. The edge categories are based on presence/absence of an edge in the actual graph, whether the edge is inter-cluster or intra-cluster for community-structured graphs, and whether the endpoints of the edge are both low-degree or both high-degree or one low-degree and one high-degree node for the BA graph. This makes a total of 2 categories for ER graphs and RGGs, 4 for PPMs, SBMs and KC, and 6 for the BA graph.

We generate 10 instances of each graph type described above, with 10 sparse-spectrum graph signals for each graph, making it a total of $N = 100$ sparse-spectrum signals for each graph type. The signals are linear combinations of $s = 5$ ($s = 2$ for KC) eigenvectors (chosen randomly for each signal) of the Laplacian matrices of the graphs. We use a measurement matrix of size 50×100 or 17×34, whose entries are i.i.d. $\mathcal{N}(0, 1)$. Furthermore, i.i.d. $\mathcal{N}(0, \sigma^2)$ noise is added to each measurement, with $\sigma = \beta \frac{\|\boldsymbol{\Phi} \boldsymbol{x}^*\|_1}{m}$, with $\beta \in \{0, 0.01, 0.02, 0.05\}$. Corresponding to each signal, a nominal graph is generated as described above.

The algorithm GES is compared with the baseline algorithm NGFT-LASSO-CV (Eqn. 6) and the ideal algorithm AGFT-LASSO-CV (Eqn. 5) using the RRMSE (Relative Root Mean-Square Error) metric, defined as $\texttt{RRMSE}(\hat{\boldsymbol{x}}) = \frac{\|\hat{\boldsymbol{x}} - \boldsymbol{x}^*\|_2}{\|\boldsymbol{x}^*\|_2}$, where $\hat{\boldsymbol{x}}$ is the estimated signal, and $\boldsymbol{x}^*$ is the ground-truth signal. The set $\Gamma := \{10^{-3+\frac{6}{19}t} : t \in \{0, 1, 2, \ldots, 19\}\}$ (i.e. 20 values between 0.001 and 1000, evenly spaced in logarithm) for all algorithms. For GES, the loss improvement factor is set to $\tau = 0.99$, and the maximum number of iterations to $d_0 = 2d$, for each $d \in \{1, 2, 5, 10\}$. In practice, the algorithm stops much earlier than the max number of iterations due to the CV error either increasing in an iteration or being close to the noise variance. This is also evidenced by the fact that the total number of edge perturbations reported by GES (Table 2, sum of correctly detected and spuriously reported edge perturbations) is mostly close to d and much smaller than d_0.

The GES algorithm is run with the following practical modifications: (1) grid search over Γ to estimate the LASSO regularization parameter μ is performed only for the candidate graph at the end of a step and is re-used for the perturbed graphs in the next step in order to speed up the experiments; (2) K-fold (or

Graph	# Edges Perturbed	AGft-Lasso-Cv	NGft-Lasso-Cv	Ges
PPM	1	0.0000	0.1111	0.0003
	2		0.2125	0.0156
	5		0.3410	0.1618
	10		0.4631	0.3856
SBM	1	0.0000	0.0372	0.0001
	2		0.0685	0.0009
	5		0.1529	0.0273
	10		0.2259	0.1277
ERG	1	0.0000	0.0843	0.0000
	2		0.1304	0.0024
	5		0.2343	0.1082
	10		0.3218	0.2634
RGG	1	0.0000	0.0357	0.0000
	2		0.0855	0.0000
	5		0.1508	0.0248
	10		0.2580	0.1589
BAG	1	0.0000	0.0472	0.0000
	2		0.1066	0.0003
	5		0.1788	0.0246
	10		0.2657	0.1772
KCG	1	0.0001	0.0982	0.0001
	2		0.1627	0.0197
	5		0.2865	0.1379
	10		0.4266	0.3390

Table 1: RRMSE of signal recovered from noiseless measurements ($m = 50, n = 100$ except for KCG where $m = 17, n = 34$) via Greedy Edge Selection (Ges), Lasso with the nominal graph (Ngft-Lasso-Cv), and Lasso with the actual graph (Agft-Lasso-Cv), for various number of perturbed edges.

multi-fold) cross-validation (with $K = 5$) is used instead of single-fold CV to improve accuracy since the total number of available measurements is small – in K-fold cross-validation, the CV error is computed K times, each time with a different subset of the m measurements, each subset of size $\lfloor \frac{m}{K} \rfloor$ (with reconstruction done on the remaining $m - \lfloor \frac{m}{K} \rfloor$ measurements), and the sum of the K CV errors is returned; and (3) the prior set of possible faulty edges Ω (of size $|\Omega| = 100$ as specified earlier) is provided to the Ges so that only these edge perturbations are tried, which speeds up the experiments. The latter is justified since in real-world applications, such prior information may be available. For example, in a CT graph, we could conclude that two individuals may have come in contact if their phones use the same WiFi access points, even though Bluetooth-based contact tracing may suggest otherwise (e.g. by being located in the same office room at a far-enough distance).

Results: The RRMSE of the Greedy Edge Selection (Ges) algorithm for recovery of sparse-spectrum signals from noiseless measurements ($\sigma = \beta = 0$) is presented in Table. 1, for each graph model mentioned earlier, with $d = \{1, 2, 5, 10\}$ edges perturbed to create the nominal graph. We see that for each graph type, Ges outperforms the baseline method of recovery using the nominal graph, i.e. Ngft-Lasso. As expected, the RRMSE increases as the number of perturbations are increased. For one or two edge perturbations, the RRMSE is close to zero or very small, whereas for upto five perturbations, the RRMSE is at most half of that obtained using Ngft-Lasso, and often much lower than that. Even for ten edge perturbations, the RRMSE of Ges is significantly lower than that of Ngft-Lasso.

Table 2 presents the fraction of cases in which Ges successfully recovers (i.e. recovers with *zero* error) the actual graph from the nominal graph. We see that for most graph types, the actual graph is recovered in close to 100% cases when up to two edges are perturbed. Despite being a greedy algorithm, the actual

Graph	# Edges Perturbed	Frac. Actual Graph Recovered	Mean # Edges Detected	Mean # Spurious Edges
PPM	1	1.00	1.00	0.00
	2	0.93	1.89	0.07
	5	0.37	2.60	1.58
	10	0.00	1.03	3.06
SBM	1	0.90	0.90	0.00
	2	0.78	1.75	0.05
	5	0.43	3.62	1.36
	10	0.02	3.47	3.54
ERG	1	1.00	1.00	0.00
	2	0.98	1.96	0.02
	5	0.43	2.93	1.41
	10	0.01	1.52	2.99
RGG	1	0.98	0.98	0.00
	2	0.96	1.96	0.00
	5	0.62	3.94	0.95
	10	0.03	2.55	3.80
BAG	1	0.97	0.97	0.00
	2	0.92	1.91	0.02
	5	0.52	3.92	1.06
	10	0.04	2.78	3.93
KCG	1	0.89	0.91	0.14
	2	0.66	1.42	1.00
	5	0.04	0.94	3.26
	10	0.00	0.95	4.19

Table 2: Fraction of actual graphs successfully recovered, mean number of edge perturbations correctly detected, and mean number of spurious edge perturbations reported by GES, for various number of perturbed edges on different graph models ($m = 50, n = 100$ except for KCG where $m = 17, n = 34$).

graph gets recovered in a significant fraction of cases even when five edges perturbations were performed. In some cases the actual graph is recoverable via GES even if ten edge perturbations were induced. Table 2 also shows the average number of edge perturbations *correctly* detected by GES, and the average number of spurious edge perturbations reported by it. Close to 100% of edge perturbations are correctly detected for upto two edge perturbations made to the actual graph, and around 50% to 80% for most graphs when five edge perturbations are made. The number of spurious edge perturbations reported by GES is close to zero for upto two edge perturbations, and small for five edge perturbations. With ten edge perturbations, the number of edge perturbations correctly detected drops and the number of spurious edge perturbations reported increases. But it is worth noting that the problem of edge perturbation recovery from *compressive measurements* of a graph signal is quite challenging due to the eigenvectors of the nominal graph being significantly perturbed for even a small number of edge perturbations. For example, note that even a small number of edge perturbations (as low as 1-3) can seriously affect signal recovery – see Table 1 for the PPM graph when 2 and 5 edges are perturbed, where NGFT-LASSO-CV produces very high errors of 0.21 and 0.34, but GES reduces them to 0.016 and 0.16, respectively. Algorithms such as GES provide efficient ways of signal recovery in compressive settings in the face of adverse conditions brought in by (even a small number of) edge perturbations. For comparison, the work in Ceci & Barbarossa (2020) considers recovery of upto four edge perturbations from a known, *uncompressed* graph signal with Gaussian priors. For analysis of edge recovery by category (intra- or inter-cluster), see Sec. S.IV of the supplemental material.

The performance of GES with noisy measurements is shown in Table 3, for the PPM. We find that the GES algorithm performs well even in the presence of noise. Its RRMSE is significantly less than that of NGFT-LASSO-CV for upto 5 edge perturbations, and is comparable to that of AGFT-LASSO-CV when the

Signal Model	Noise Level (β)	# Edges Perturbed	AGft-Lasso-Cv	NGft-Lasso-Cv	Ges
Sparse-Spectrum	0.01	1	0.0117	0.1125	0.0082
		2		0.2113	0.0145
		5		0.3439	0.1772
		10		0.4931	0.4355
	0.02	1	0.0223	0.1192	0.0156
		2		0.2137	0.0247
		5		0.3457	0.1792
		10		0.4940	0.4441
	0.05	1	0.0539	0.1412	0.0396
		2		0.2351	0.0579
		5		0.3555	0.2180
		10		0.4987	0.4464
Band-Limited	0.01	1	0.0114	0.0131	0.0092
		2		0.0141	0.0096
		5		0.0185	0.0137
		10		0.0238	0.0191
	0.02	1	0.0227	0.0243	0.0179
		2		0.0248	0.0191
		5		0.0278	0.0233
		10		0.0320	0.0266
	0.05	1	0.0560	0.0559	0.0438
		2		0.0551	0.0446
		5		0.0586	0.0484
		10		0.0598	0.0497

Table 3: Performance of Ges with measurement noise of different levels given by $\beta \in \{0.01, 0.02, 0.05\}$ ($m = 50, n = 100$).

number of edge perturbations is 1 or 2. Occasionally, we find that Ges is able to do slightly better than Agft-Lasso-Cv. We suspect that this is because only 20 different values were tried during grid search for the Lasso regularization parameter μ, due to which Ges could find a better graph than the actual graph for the given values of μ for the purpose of signal recovery. A more fine-grained grid search would lead to the expected behaviour of AGft-Lasso-Cv performing better than Ges.

Comparison between Bfgs and Ges: In Table 4, we now present comparative performance statistics for Ges versus Bfgs for 3 perturbations induced on the KC graph of $n = 34$ nodes with $m = 17$ compressive measurements. We used an 8-fold CV, 20 values of μ used for each graph, and a prior list of 10 possibly faulty edges provided to the algorithms (to make the brute force search more manageable). In total, we perturbed 3 edges from the set of the faulty edges, and nominal graphs were created by adding inter-cluster edges to the actual graph. Average RRMSE results were computed for 100 signals. We also report the fraction of times that graph recovery was successful (all edge perturbations correctly recovered). The last two rows of Table 4 present additional results for the same settings as above, but for different methods of choosing μ: (1) For Bfgs, a single μ for the Lasso estimator was chosen using CV on the nominal graph, and Bfgs was performed using only that chosen value of μ. (2) For Ges, the value of μ was chosen at the beginning of each t^{th} iteration using CV with the GFT basis of the current graph, i.e. $\mathcal{G}^{(t-1)} := (\mathcal{V}, \mathcal{E}^{(t-1)})$, and then used for all candidate perturbations in the t^{th} iteration. From Table 4, we see the superior performance of Bfgs over Ges for both settings as expected (albeit for greater computational cost). Both algorithms significantly lower the RRMSE compared to the baseline of using the nominal graph GFT for recovery (0.0196 and 0.0287 for Ges and 0.0006 and 0.0028 for Bfgs, compared to 0.1874 for Ngft-Lasso-Cv).

Case of repeated eigenvalues: On randomly generated PPM graphs, we observed that around 96% of the eigenvectors had eigenvalues with multiplicity of 1. About 0.8% had a multiplicity of 2, and about 0.7% had a multiplicity of 3 and the remaining had a multiplicity greater than or equal to 4 (Table 5). All these

Method	RRMSE	Frac. Graphs Recovered
AGFT-LASSO-CV	0.0001	-
NGFT-LASSO-CV	0.1874	0
GES	0.0196	0.79
BFGS	0.0006	0.99
GES, μ chosen on current candidate graph	0.0287	0.75
BFGS, μ chosen on nominal graph	0.0028	0.94

Table 4: Comparison of reconstruction performance of GES and BFGS on the KC Graph with $n = 34$ nodes and using $m = 17$ compressive measurements, given $d_0 = 3$ inter-cluster edge perturbations. RRMSE results are avg. over 100 signals.

Graph type	Multiplicity	% of eigenvectors with this multiplicity
PPM with 100 nodes	1	95.998
	2	0.790
	3	0.696
	≥ 4	2.516

Table 5: Eigenvalue multiplicities for planted partition model graphs with $n = 100$ nodes, computed over 1000 graphs.

percentages were computed across 1000 instances each of different types of synthetic graphs with $n = 100$ nodes. Two eigenvalues were considered equal if the absolute difference between them was less than or equal to $\epsilon = 10^{-8}$. For the same experiments on the SBM and RGG graphs, very similar results were obtained and are hence not shown separately – more than 98% and 97% of the eigenvectors, respectively, had eigenvalue multiplicities equal to 1 for these graphs. For the BA and ER graphs, all eigenvectors had a eigenvalue multiplicity of 1. The good reconstruction results shown in Tables 1 and 2 confirm the hypothesis that signal reconstruction is not adversely affected if the (randomly generated and sparse) signal support does not cover eigenvectors associated with eigenvalues having multiplicity greater than one, or if the multiplicity is not too much larger than 1. Furthermore, we also computed the effective sparsity of the signals in the GFT bases, i.e. the sum of multiplicities of the eigenvector groups which form the signal support. We observed empirically that the effective sparsity was equal to the true sparsity (i.e. ℓ_0 norm of the coefficient vector) in majority of the test cases, and very rarely greater than the true sparsity by more than 5. Further details are now included in the supplemental material in Sec. S.VIII.

Using closed-form approximations for eigenvector perturbation: Table 6 shows the RRMSE obtained using the GES-AE algorithm, in which the eigenvector perturbation approximation Ceci & Barbarossa (2020) (Eqn. 13) is used in place of eigendecomposition in GES (see Sec. 3.4), for compressive recovery of sparse-spectrum and band-limited signals on PPMs with a few edge perturbations. We find that for both

Signal Model	# Edges Perturbed	AGFT-LASSO-CV	NGFT-LASSO-CV	GES	GES-AE
Sparse-Spectrum	1	0.0000	0.1142	0.0000	0.1109
	2		0.2175	0.0061	0.2306
	5		0.3492	0.1821	0.3335
	10		0.5006	0.4368	0.5139
Band-Limited	1	0.0000	0.0042	0.0001	0.0033
	2		0.0059	0.0003	0.0054
	5		0.0130	0.0024	0.0135
	10		0.0195	0.0104	0.0202

Table 6: Performance of Greedy Edge Selection with approximated eigenvectors (GES-AE) compared to that of GES, AGFT-LASSO-CV, and NGFT-LASSO-CV for sparse-spectrum and bandlimited signals, all with $m = 50, n = 100$.

sparse-spectrum and band-limited signals, the RRMSE of GES-AE is significantly higher than that for GES, across a range of edge perturbations. Moreover, the RRMSE of GES-AE is close to that obtained with the

Method	d	RRMSE	Frac. of Graphs Recovered
AGFT-LASSO-CV	1	0.0292	-
NGFT-LASSO-CV	1	0.2126	0
GES	1	0.0688	0.83
AGFT-LASSO-CV	2	0.0292	-
NGFT-LASSO-CV	2	0.2956	0
GES	2	0.0860	0.74
AGFT-LASSO-CV	5	0.0292	-
NGFT-LASSO-CV	5	0.4719	0
GES	5	0.3384	0.22
AGFT-LASSO	10	0.0292	-
NGFT-LASSO	10	0.5631	0
GES	10	0.5403	0.02

Table 7: RRMSE results for signal interpolation over PPM graph perturbed by d edges, for signals with a sparse representation in the actual GFT, using GES algorithm.

nominal graph, and sometimes slightly worse. Hence we do not find the eigenvector perturbation approximation (a first order approximation) to be useful when applied to greedy edge selection with cross-validation. Perhaps a better approximation with higher order terms in it might be more useful. A major reason for the performance of GES-AE could be that the condition under which the approximation is applicable – given by Eqn. 14 – gets violated often. We deem the condition to be violated for an edge $\{i, j\}$ and the k^{th} eigenvector $\boldsymbol{v}_k$ if $(\boldsymbol{v}_k(i) - \boldsymbol{v}_k(j))^2 > 4(\lambda_{k+1} - \lambda_k))$ for edge addition or $(\boldsymbol{v}_k(i) - \boldsymbol{v}_k(j))^2 > 4(\lambda_k - \lambda_{k-1})$ in case of edge deletion. In our experiments, the fraction of times this condition was violated for at least one eigenvector out of the first five (i.e. the band-limited eigenvectors) for a PPM graph with $n = 100, l = 5, p = 0.9, q = 0.01$ was 87.7%, if an inter-cluster edge is perturbed. Similarly, we found that the condition in Eqn. 14 was violated 74.7% of the times for at least one eigenvector out of five for a random choice of five eigenvectors and any kind of edge. While this condition does not get violated for intra-cluster edges in the band-limited case, the amount of perturbation in the low-frequency eigenvectors by such edges is very small, such that the RRMSE for these cases is close to zero even for NGFT-LASSO-CV, and hence these cases do not contribute much to the average RRMSE.

Signal Interpolation on Perturbed Graphs: We now present results for interpolation of signals defined on the nodes of a perturbed graph, assuming that the signal is sparse in the GFT basis of the actual graph. In our experiments, we use signal values defined at only $m = 50$ nodes out of a total of $n = 100$ nodes. The signal coefficient vectors contain only 5 non-zero elements. The nominal graph contains $d \in \{1, 2, 5, 10\}$ edge perturbations w.r.t. the actual graph. The values at the remaining nodes are obtained via the GES algorithm. Numerical results for this experiment are presented in Table 7 for a PPM graph, with results averaged over 100 instances. The results show substantially improved performance with GES for $d \in \{1, 2\}$ and some improvement for $d = 5$.

5.2 Inferred Linear-Edge Compressive Image Recovery

Experiment Setup: For evaluation of ILECIR, we simulate the patch-wise compressive acquisition of 10 images each from the following datasets: (1) A synthetic dataset we created, containing piece-wise smooth images in the form of a union of regions individually containing intensity values expressed by polynomials of x, y; (2) The Berkeley segmentation dataset[3], a set of natural images used in image segmentation tasks; (3) The Tom and Jerry dataset[4], a dataset of still frames from the popular Tom and Jerry cartoon; and (4) The NYU Depth dataset[5], a dataset of depth-map images of indoor scenes. While ground-truth segmentation is available for the synthetic dataset, the available segmentations for the other datasets are either human-

[3] https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/

[4] https://www.kaggle.com/datasets/balabaskar/tom-and-jerry-image-classification

[5] https://cs.nyu.edu/~silberman/datasets/nyu_depth_v2.html

labelled or machine-generated. In each dataset, the segmentation information is used to construct the actual graph, by dropping the edges of the lattice graph which connect pixels in two different segments of a patch.

Patch-wise compressive image acquisition is simulated for an image in the following manner. The image is divided into non-overlapping patches of size 8×8. Each patch is arranged in row-major order to produce a vector of $n = 64$ dimensions, which forms the signal $\boldsymbol{x}^*$. Measurements are generated using Eqn. 1, with a $m \times n$ measurement matrix whose entries are i.i.d. $\mathcal{N}(0, 1)$, with $m \in \{20, 30, 40\}$. Furthermore, i.i.d. $\mathcal{N}(0, \sigma^2)$ noise is added to each measurement, with $\sigma = \beta \frac{\|\boldsymbol{\Phi}\boldsymbol{x}^*\|_1}{m}$, with $\beta \in \{0, 0.01, 0.02, 0.05\}$. These measurements are then decoded using the ILECIR algorithm, and the 8×8 patch is estimated. For gray-scale images, the estimated values are clipped to be between 0 and 255 and rounded to the nearest integer. For depth maps, negative values are set to 0. The recovered patches are stitched together to reconstruct the image. The results of ILECIR are compared to those with DCT-based recovery (DCT-LASSO-CV), and using the GFT of the ground-truth segmentation (SEGGFT-LASSO-CV) in terms of the RRMSE and Structural Similarity Index Measure (SSIM)[6] of the recovered images relative to the ground truth. For the algorithms, $\Gamma \triangleq \{10^{-3+\frac{6}{19}t} : t \in \{0, 1, 2, \ldots, 19\}\}$, $K = 5$ folds are used for cross-validation, and the loss improvement factor τ is set to 0.99. To speed up ILECIR, grid search over Γ is performed only for DCT-based recovery, and the same value of μ is re-used for recovery with the image-edge partitioned graphs. However, after an image-edge is chosen, μ is re-computed using CV to yield the final patch estimate.

Quantitative comparison in the noiseless setting: Fig. 2 shows the RRMSE and SSIM of images recovered using ILECIR, DCT-LASSO-CV and SEGGFT-LASSO-CV from $m \in \{20, 30, 40\}$ linear measurements, for 10 images each from the aforementioned datasets. Also see Tables S.VI-S.XVII of the supplemental for the numerical values. We note that ILECIR substantially improves over DCT-LASSO-CV in terms of the RRMSE and SSIM of the recovered images on the Synthetic and the NYU Depth datasets. It also shows some improvement for the images in the Tom and Jerry dataset. For Tom and Jerry and the NYU Depth datasets, the improvement is most pronounced when $m = 20$ linear measurements are used. Such improvement is expected since these are piece-wise smooth images – while the smooth regions are accurately recovered by DCT-LASSO-CV, the patches containing sharp edges are not recovered very accurately. ILECIR performs better on such patches, while maintaining the accuracy afforded by DCT-LASSO-CV on smooth patches. The SEGGFT-LASSO-CV method has near-perfect recovery for the Synthetic dataset. This substantiates the rationale behind using the GFT basis of the 2-D lattice graph partitioned according to the segments of a patch for recovery. Since ILECIR only models linear image edges and does not model more than one image edge in a patch, recovery using it is not perfect. Notably, ILECIR often performs better than the SEGGFT-LASSO-CV method for the Tom and Jerry and NYU Depth Datasets. This means that it is able to perform edge detection slightly better than the algorithm used to perform edge detection for the images from the Tom and Jerry dataset. For the NYU Depth Dataset, we found that the provided segmentation maps were not properly aligned with the depth map images (see Fig. 3 for an example of an incorrectly labelled segmentation map of a patch from this dataset). Due to this, the SEGGFT-LASSO-CV method performs *worse* than DCT-LASSO-CV for the NYU Depth dataset. This demonstrates a strength of the ILECIR method – since the underlying segmentation is automatically inferred, it circumvents possible errors in the segmentation map if it were available.

RRMSE and SSIM of images recovered using ILECIR are higher than those recovered using DCT-LASSO-CV for some images of the Berkeley dataset. Since natural images contain vast regions of detailed texture and not just gradient of intensity, it is hard to improve upon the baseline DCT-LASSO-CV method by modelling just a single linear edge. This is also substantiated by the performance of the SEGGFT-LASSO-CV method, which is similar to DCT-LASSO-CV and ILECIR. However, a visual inspection of the images recovered by ILECIR, SEGGFT-LASSO-CV and DCT-LASSO-CV would reveal that ILECIR and SEGGFT-LASSO-CV perform better along the edges in these images (see later in this section).

Qualitative Comparison in the noiseless setting: The results of image recovery via SEGGFT-LASSO-CV, DCT-LASSO-CV, and ILECIR for one image of each dataset are shown in Fig. 4, along with the original image in each case. A careful inspection of the images reveals that in each case, the images recovered by ILECIR have higher fidelity than the corresponding image recovered by DCT-LASSO-CV along sharp edges

[6] `https://en.wikipedia.org/wiki/Structural_similarity`

in the image. The quality of the recovered images remains the same in regions where there are no sharp edges. In Fig. 4 (Row 1), for the image from the Synthetic dataset, we clearly see that ILECIR has good recovery in regions where there is only a single edge which is close to linear. In contrast, DCT-LASSO-CV produces many artifacts along the edges. The image recovered by ILECIR also possesses some artifacts along the edges, but much fewer than the one recovered by DCT-LASSO-CV. These artifacts may appear for ILECIR whenever the ground-truth image edge in the patch does not exactly match any of the image edges that ILECIR tests for. ILECIR does not do well around corners, due to the presence of more than one image edge in the patch. However, a careful inspection reveals that for some patches, the recovery quality for ILECIR seems to be better than DCT-LASSO-CV even in the presence of two edges, and there does not appear to be a case where ILECIR has worse recovery quality than DCT-LASSO-CV. This is due to the high-probability RMSE improvement guarantee of ILECIR (due to cross-validation) as established by Theorem 3.3.

In Fig. 4 (Row 2), there are much fewer artifacts around the face of the polar bear in the image recovered by ILECIR as compared to the one recovered by DCT-LASSO-CV. In the same region, we also see that while there are fewer artifacts for SEGGFT-LASSO-CV compared to DCT-LASSO-CV, the recovery by ILECIR is much better than for SEGGFT-LASSO-CV. This once again highlights the strength of ILECIR compared to using human-labelled segmentation (which may contain errors) for recovery and is not available during compressive recovery. In Fig. 4 (Row 3), there are much fewer artifacts for ILECIR around the ears and the head of Tom (the cat), compared to for DCT-LASSO-CV. However, there are many artifacts on the face of Tom for both ILECIR and DCT-LASSO-CV. This highlights the inability of ILECIR to improve upon DCT-LASSO-CV in regions containing many edges. In Fig. 4 (Row 4), we observe the same pattern – ILECIR performs better than DCT-LASSO-CV and even SEGGFT-LASSO-CV around the edges in the depth map. The reconstruction by each algorithm on an individual patch of an image from the NYU Depth dataset is also shown in Fig. 3, which clearly demonstrates the recovery of the ground-truth segmentation and the original patch using ILECIR.

Edgemap Recovered by Ilecir: Since the ILECIR algorithm discovers the best linear image edge for each patch, an edge-map of the original image may be recovered by stitching together the edge-maps of each patch. Fig 5 shows an image from the synthetic dataset (col 1), and its edge-map (col 2) recovered by the ILECIR algorithm. The original edges are correctly detected for most patches.

Performance of Greedy Edge Selection on Compressive Image Recovery: We test the effectiveness of GES on the problem of patch-wise compressive image recovery on an image from the Synthetic dataset. Fig. 6 and Table 8 show the RRMSE and SSIM of the recovered images when GES is used for recovery, with a maximum of $d_0 \in \{1, 2, 5, 10\}$ edges of the 2-D lattice graph of a patch being allowed to be perturbed by the algorithm. We see that using GES (especially when $d_0 = 10$ graph edges are allowed to be perturbed) is marginally better than using DCT-LASSO-CV – however, it is much better to use ILECIR. This is evident in Fig. 6, as well – a careful inspection reveals that the image recovered by GES-10 ($d_0 = 10$) has slightly fewer artifacts than the one recovered by DCT-LASSO-CV, but recovery via ILECIR (see Fig. 4) is much better. Fig. 5 shows the edgemap recovered by various algorithms, including GES-5 and GES-10 (cols 3 and 4). An edgemap for GES is recovered in the following manner – whenever the GES algorithm drops a (2-D lattice graph) edge between two pixels, the right pixel or the bottom pixel of the graph edge is marked with white color (denoting a pixel belonging to an image edge), and remaining pixels are marked with black color (denoting background pixels). We see that the edgemap recovered by GES follows the image edges very coarsely. It consists of many isolated dots, instead of groups of pixels connected as an edge. In comparison, the edgemap recovered by ILECIR consists of linear edges in each patch. The ILECIR algorithm takes advantage of the structure in perturbations of the graph edges, and is able to outperform GES, which perturbs graph edges in an unstructured manner.

Effect of Noise on Ilecir recovery performance: Table 9 shows the RRMSE and SSIM of images recovered using ILECIR from noisy measurements of an image of the Synthetic dataset, with different noise levels $\beta \in \{0, 0.01, 0.02, 0.05\}$. We find that ILECIR performs significantly better than DCT-LASSO-CV even in the presence of noise. In fact ILECIR on measurements with $\beta = 0.02$ has better RRMSE and approximately the same SSIM as DCT-LASSO-CV with noiseless measurements. We also find that the RRMSE and SSIM of SEGGFT-LASSO-CV worsens significantly with measurement noise. Fig. 7 shows the images recovered using DCT-LASSO-CV and ILECIR for different noise levels. We note that the images recovered by ILECIR

# Measurements (m)	Method	RRMSE	SSIM
40	DCT-LASSO-CV	0.0335	0.9598
	SEGGFT-LASSO-CV	0.0005	1.0000
	ILECIR	0.0161	0.9938
	GES-1	0.0335	0.9606
	GES-2	0.0327	0.9633
	GES-5	0.0317	0.9657
	GES-10	0.0317	0.9658
30	DCT-LASSO-CV	0.0433	0.9406
	SEGGFT-LASSO-CV	0.0011	1.0000
	ILECIR	0.0253	0.9856
	GES-1	0.0440	0.9395
	GES-2	0.0430	0.9425
	GES-5	0.0421	0.9450
	GES-10	0.0419	0.9454
20	DCT-LASSO-CV	0.0656	0.8914
	SEGGFT-LASSO-CV	0.0057	0.9993
	ILECIR	0.0353	0.9744
	GES-1	0.0685	0.8845
	GES-2	0.0682	0.8860
	GES-5	0.0679	0.8868
	GES-10	0.0682	0.8863

Table 8: Performance of GES with various perturbation budgets on Synthetic Image ID 0.

Noise Level	Method	RMSE	SSIM
0	DCT-LASSO-CV	0.0335	0.9598
	SEGGFT-LASSO-CV	0.0005	1.0000
	ILECIR	0.0161	0.9938
0.01	DCT-LASSO-CV	0.0343	0.9531
	SEGGFT-LASSO-CV	0.0079	0.9929
	ILECIR	0.0187	0.9850
0.02	DCT-LASSO-CV	0.0375	0.9336
	SEGGFT-LASSO-CV	0.0156	0.9730
	ILECIR	0.0240	0.9620
0.05	DCT-LASSO-CV	0.0520	0.8349
	SEGGFT-LASSO-CV	0.0368	0.8739
	ILECIR	0.0468	0.8457

Table 9: RRMSE and SSIM of ILECIR, DCT-LASSO-CV and SEGGFT-LASSO-CV for Image ID 0 of the Synthetic dataset for different noise levels ($\beta \in \{0, 0.01, 0.02, 0.05\}, m = 40, n = 64$).

are of better quality than the corresponding images recovered by DCT-LASSO-CV, across all noise levels. The image recovered by ILECIR with noise level $\beta = 0.02$ looks better near the edges in the image than the one recovered by DCT from noiseless measurements.

Dealing with repeated eigenvalues: For ILECIR, the nominal graph is always a connected grid graph. However, the graphs that subsequently evolve in the ILECIR contain two (disconnected) components separated by an image edge. For each such segmentation, an adjacency matrix is created by dropping the edges of the 2-D lattice graph whose two endpoints lie in two different segments. We computed the eigenvectors of Laplacian matrices of the two segments in each patch separately. Note that the eigenvectors of the Laplacian matrix of a graph with more than one connected component are the same as the eigenvectors of each component, with the entries corresponding to nodes of the remaining components set to zero. This fact helped us obtain the eigenvectors of the Laplacian matrix of the complete segmented patch. This procedure is helpful to avoid introducing correlations between the two segments, which may happen in case there are repeated eigenvalues. Notably, the eigenvalue of 0 gets repeated k times if there are k connected components in a graph. These computed GFTs are used in the ILECIR algorithm.

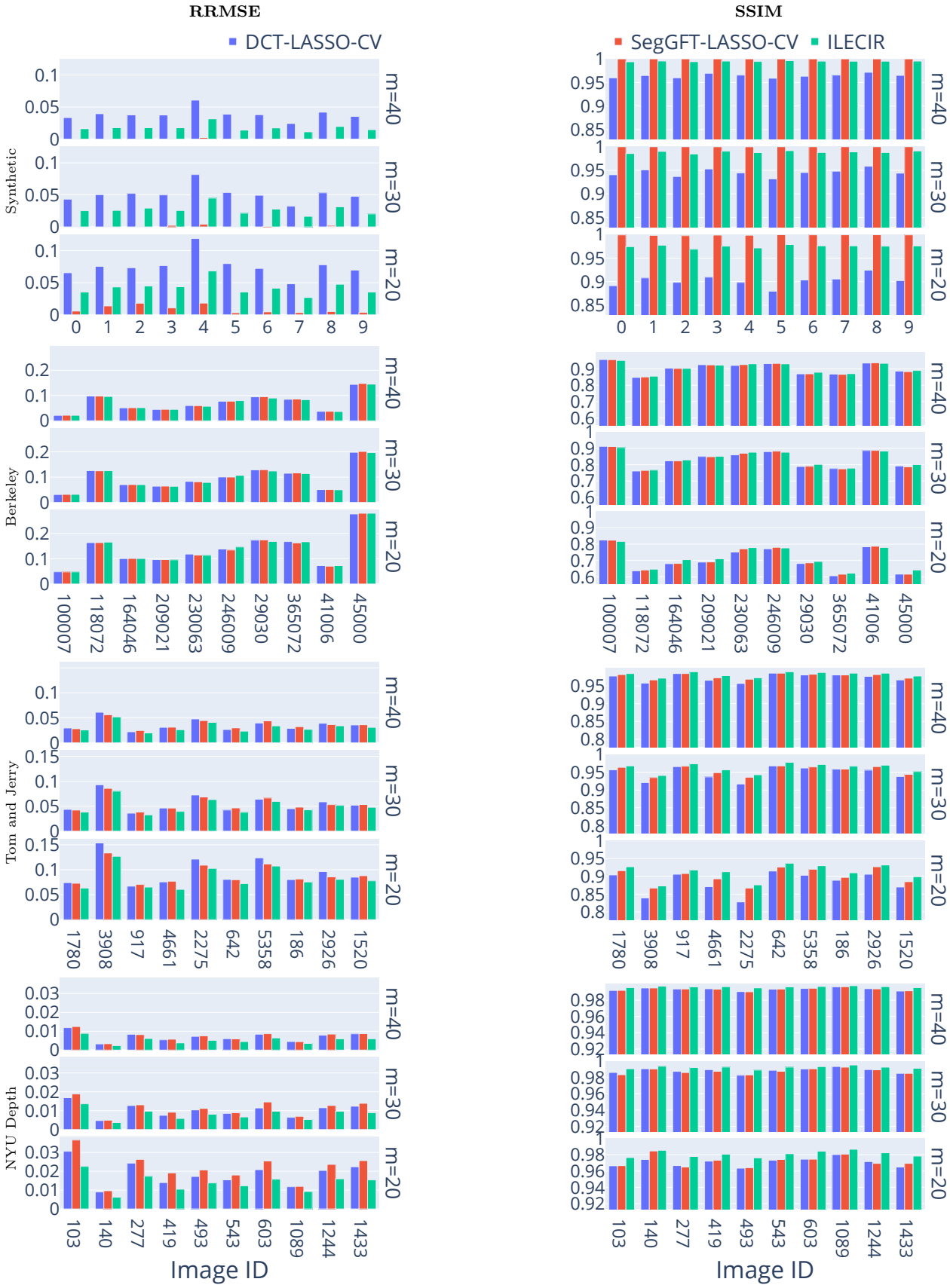


Figure 2: RRMSE and SSIM using ILECIR, DCT-LASSO-CV and SEGGFT-LASSO-CV (Segmentation-aware recovery) in the noiseless regime, with $n = 64$ for 8×8 patches. Also see Tables S.VI to S.XVII of the supplemental.

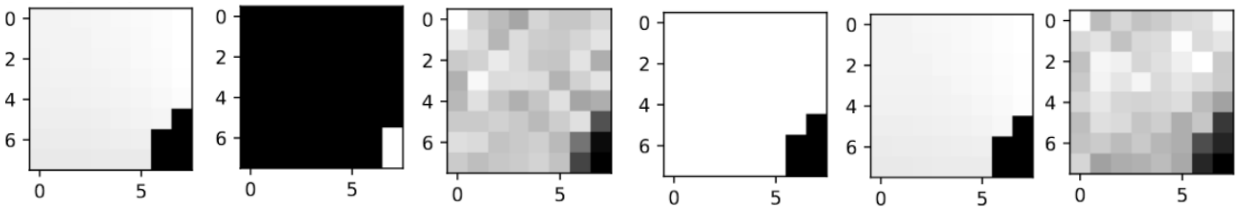


Figure 3: Left to right: An 8×8 patch from an image in the NYU depth dataset (leftmost), its *incorrectly* labelled segmentation (second from left), signal recovery by SegGFT-Lasso-Cv (poor quality), recovery of correction segmentation by Ilecir, recovery of patch by Ilecir, patch recovered via Dct-Lasso-Cv. Note: the two regions of the segmentation maps are colored arbitrarily.

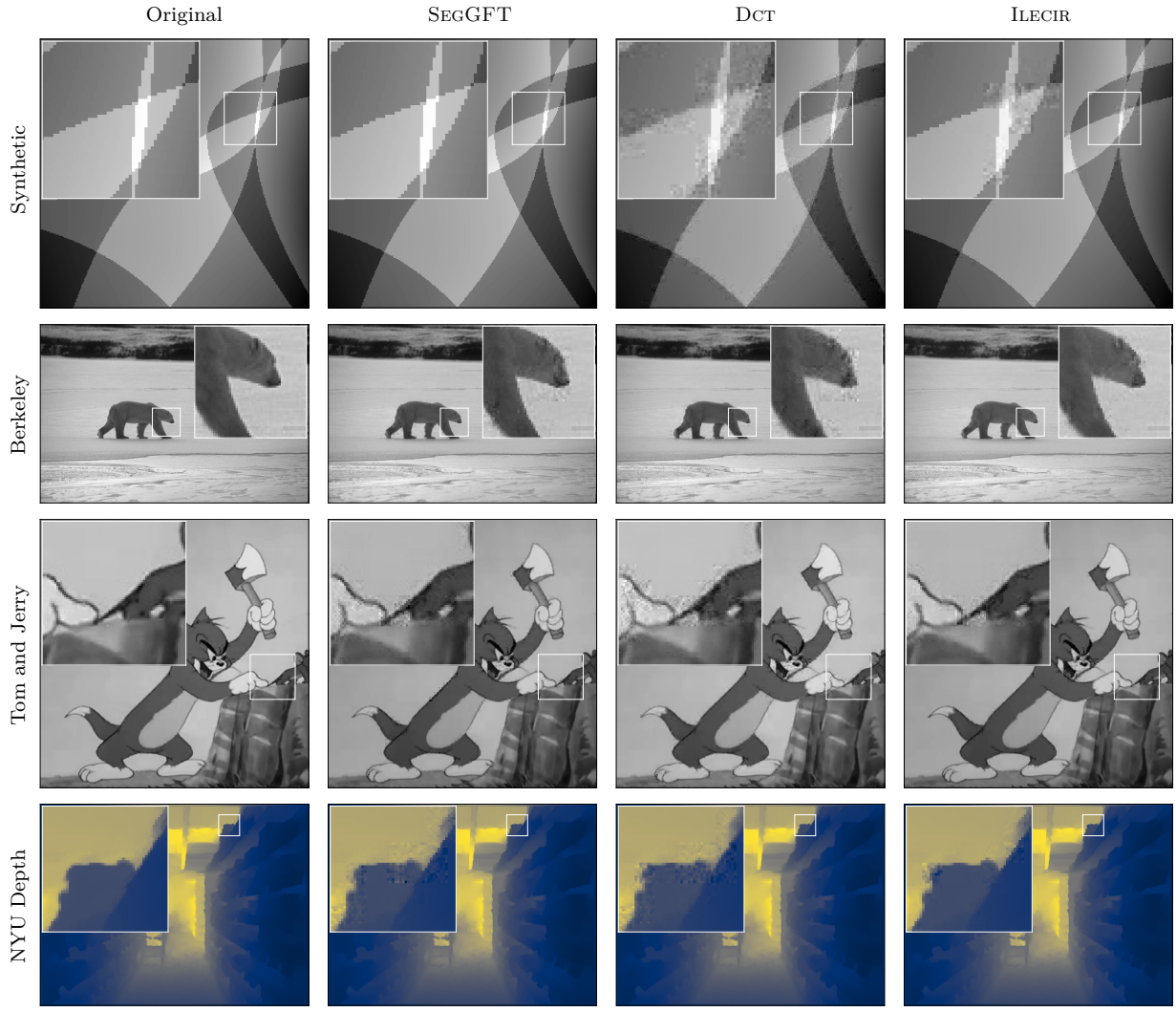


Figure 4: Zoomed view of images from the Synthetic (top row), Berkeley (second row), Tom and Jerry (third row), and NYU Depth (bottom row) datasets, recovered patch-wise from $m = 40$ compressive measurements per 8×8 patch via various algorithms in the noiseless setting. The columns denote, from left to right: Col 1 – the original image, images recovered via SegGft-Lasso-Cv (Col 2), via Dct-Lasso-Cv (Col 3), via Ilecir (Col 4). Note that Ilecir produces significantly smoother reconstruction of the edges (the polar bear's face against the snow, Tom's finger against the wall, the object against the background in the depth map) than Dct-Lasso-Cv. Zoom into PDF for a clearer view. Enlarged images are provided in the supplementary materials (Figs. S.3 to S.6).

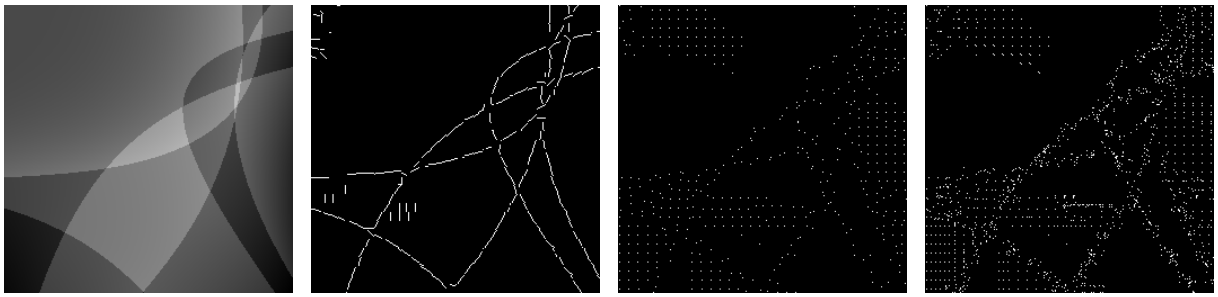


Figure 5: A synthetic image (col 1), and its edgemap as recovered patch-wise from $m = 30$ compressive measurements per 8×8 patch via the ILECIR (col 2), GES-1 (col 3), and GES-10 (col 4) algorithms.

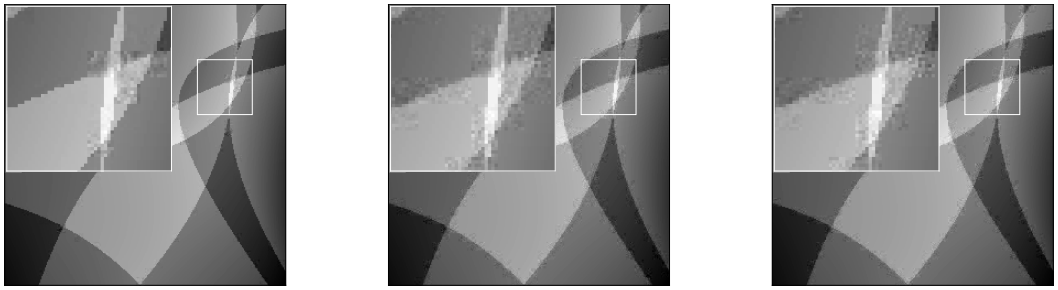


Figure 6: Comparison of images recovered using (from left to right) ILECIR, DCT-LASSO-CV, and GES-10. GES shows only very minor improvement over DCT, such as in the top-right corner of the zoomed portion, whereas ILECIR performs better reconstruction in areas with a single image edge. Hence ILECIR is preferred for edge-aware compressive image recovery.

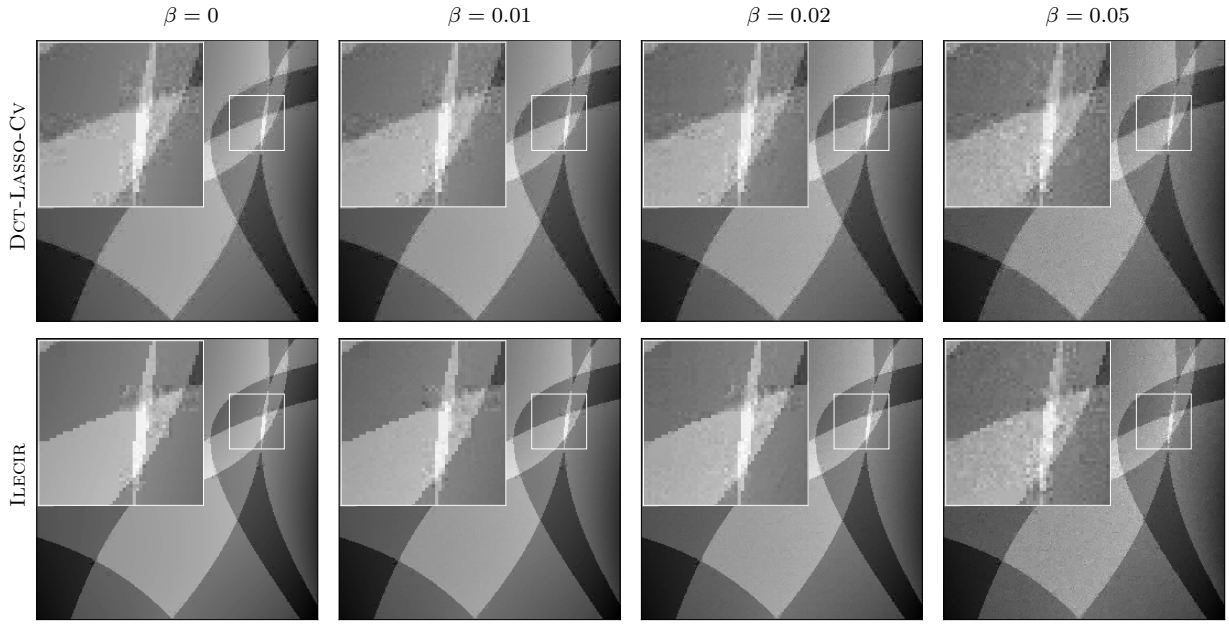


Figure 7: Zoomed view of Image ID 0 of the Synthetic dataset, recovered patch-wise from $m = 40$ compressive measurements per 8×8 patch using the DCT-LASSO-CV (top row) and the ILECIR (bottom row) algorithms for various noise levels ($\beta \in \{0, 0.01, 0.02, 0.05\}$). Note the significantly smoother edge reconstruction and edge artifact reduction in case of ILECIR as compared to DCT-LASSO-CV.

6 Conclusion and Future Work

This work proposes a new technique of compressive graph signal recovery in cases where the graph topology is partly incorrectly specified. This is a novel computational problem (to our best knowledge), with no existing literature that targets this specific problem. At the core of the technique to correct for errors in graph topology, lies the concept of cross validation in compressed sensing. A novel algorithm is proposed and supportive theoretical results are stated and proved. A large number of computer simulations are presented, including results on compressive image recovery where images or image patches are modeled as lattice graphs. Since intensity values at pixels/nodes that lie on opposite sides of an image edge are uncorrelated, the presence of a graph edge that links such nodes is regarded as a perturbation that needs to be corrected. Thus edge-preserving compressive recovery is cast quite interestingly as a graph perturbation problem. This specific idea has been used in image compression before Hu et al. (2014), where the application innately has access to the complete graph as well as the underlying image. However its application to compressive image reconstruction (as in our work) is quite novel – in our case, the underlying image as well as the underlying image graph are both unknown, and both these are estimated from just the compressive measurements. Moreover, our technique is naturally applicable to image data that are defined on *arbitrary non-Cartesian domains*, for examples on the vertices of a 3D model. In such cases, our perturbed GFT technique fits in very naturally, even though the eigenvectors of the Laplacian of the actual graph *may not be the DCT*. Furthermore, our technique can be easily extended to handle dropping of edges even in weighted graphs. We also note that the idea of signal sparsity in the GFT for compressive recovery has not been extensively explored in the literature, with most papers focusing on band-limited approximations Zhu & Rabbat (2012) unlike our approach. Finally, we show in the supplemental material (Sections 6 and 7) that our method may be used to recover signals which admit some other structure than sparsity in the GFT, such as being piece-wise constant, by using a different regularizer than the ℓ_1-norm of GFT, such as Graph Total Variation.

There are a few major avenues for future work listed below: (1) Research towards a more computationally efficient strategy to uncover all graph perturbations, over and above our greedy algorithms, and extending the strategy to handle weighted graphs, such as by using gradient descent of the Lasso objective over Laplacian matrices, parameterized in terms of eigenvectors and eigenvalues. One potential direction for designing computationally more efficient algorithms would be to incorporate ideas from Jayawant & Ortega (2022) to avoid repeated eigen-decompositions. (2) Improving the performance of compressive image recovery possibly using steerable bases such as those in Fracastoro et al. (2016), particularly in the presence of eigenvalues with multiplicity greater than 1, or using notions of image edge smoothness or continuity.

References

Network science. `https://en.wikipedia.org/wiki/Network_science`, 2024.

E. Candes and M. Wakin. An introduction to compressive sampling. *IEEE SPM*, 25(2):21–30, 2008.

Elena Ceci and Sergio Barbarossa. Graph signal processing in the presence of topology uncertainties. *IEEE TSP*, 68:1558–1573, 2020. doi: 10.1109/TSP.2020.2976583.

Elena Ceci, Yanning Shen, Georgios B Giannakis, and Sergio Barbarossa. Graph-based learning under perturbations via total least-squares. *IEEE TSP*, 68:2870–2882, 2020.

Yuejie Chi, Louis L. Scharf, Ali Pezeshki, and A. Robert Calderbank. Sensitivity to basis mismatch in compressed sensing. *IEEE TSP*, 59(5):2182–2195, 2011.

Xiaowen Dong, Dorina Thanou, Michael Rabbat, and Pascal Frossard. Learning graphs from data: A signal representation perspective. *IEEE SPM*, 36(3):44–63, 2019.

Marco F Duarte et al. Single-pixel imaging via compressive sampling. *IEEE SPM*, 25(2):83–91, 2008.

S. Fosson, V. Cerone, and D. Regruto. Sparse linear regression from perturbed data. *Automatica*, 122, 2020.

Giulia Fracastoro, Sophie M Fosson, and Enrico Magli. Steerable discrete cosine transform. *IEEE TIP*, 26 (1):303–314, 2016.

S. Ghosh, R. Agarwal, M. A. Rehan, S. Pathak, P. Agarwal, Y. Gupta, S. Consul, N. Gupta, Ritika, R. Goenka, A. Rajwade, and M. Gopalkrishnan. A compressed sensing approach to pooled RT-PCR testing for COVID-19 detection. *IEEE Open Journal of Signal Processing*, 2:248–264, 2021. doi: 10.1109/OJSP.2021.3075913.

Ritesh Goenka, Shu-Jie Cao, Chau-Wai Wong, Ajit Rajwade, and Dror Baron. Contact tracing enhances the efficiency of COVID-19 group testing. In *ICASSP*, pp. 8168–8172, 2021.

T. Hastie, R. Tibshirani, and M. Wainwright. *Statistical Learning with Sparsity: The LASSO and Generalizations*. CRC Press, 2015.

Wei Hu, Gene Cheung, Antonio Ortega, and Oscar C Au. Multiresolution graph fourier transform for compression of piecewise smooth images. *IEEE TIP*, 24(1):419–433, 2014.

J.D. Ianni and W.A. Grissom. Trajectory auto-corrected image reconstruction. *Magnetic Resonance in Medicine*, 76(3), 2016.

Ajinkya Jayawant and Antonio Ortega. Practical graph signal sampling with log-linear size scaling. *Signal Processing*, 194:108436, 2022.

Alexander Jung, Nguyen Tran, and Alexandru Mara. When is network LASSO accurate? *Frontiers in Applied Mathematics and Statistics*, 3:28, 2018.

Robert A Kleinman and Colin Merkel. Digital contact tracing for covid-19. *Cmaj*, 192(24):E653–E656, 2020.

Kuldeep Kulkarni et al. Reconnet: Non-iterative reconstruction of images from compressively sensed measurements. In *CVPR*, pp. 449–458, 2016.

Aris S Lalos, Iason Nikolas, Evangelos Vlachos, and Konstantinos Moustakas. Compressed sensing for efficient encoding of dense 3d meshes using model-based bayesian learning. *IEEE Transactions on Multimedia*, 19 (1):41–53, 2016.

Faisal Mahmood, Nauman Shahid, Ulf Skoglund, and Pierre Vandergheynst. Adaptive graph-based total variation for tomographic reconstructions. *IEEE SPL*, 25(5):700–704, 2018.

Lee Middleton and Jayanthi Sivaswamy. *Hexagonal Image Processing: A Practical Approach (Advances in Pattern Recognition)*. Springer-Verlag, Berlin, Heidelberg, 2005. ISBN 1852339144.

Jari Miettinen, Sergiy A Vorobyov, and Esa Ollila. Modelling and studying the effect of graph errors in graph signal processing. *Signal Processing*, 189:108256, 2021.

Jonathan M. Nichols, Albert K. Oh, and Rebecca M. Willett. Reducing basis mismatch in harmonic signal recovery via alternating convex search. *IEEE SPL*, 21(8):1007–1011, 2014.

Antonio Ortega, Pascal Frossard, Jelena Kovačević, José MF Moura, and Pierre Vandergheynst. Graph signal processing: Overview, challenges, and applications. *Proceedings of the IEEE*, 106(5):808–828, 2018.

H. Pandotra, E. Malhotra, A. Rajwade, and K. S. Gurumoorthy. Dealing with frequency perturbations in compressive reconstructions with Fourier sensing matrices. *Signal Process.*, 165:57–71, 2019.

J. Parker, V. Cevher, and P. Schniter. Compressive sensing under matrix uncertainties: An approximate message passing approach. In *ACSSC*, pp. 804–808, 2011.

Santiago Segarra, Antonio G Marques, Gonzalo Mateos, and Alejandro Ribeiro. Network topology inference from spectral templates. *IEEE TSIPN*, 3(3):467–483, 2017.

Z. Tan, Y Peng, and A Nehorai. Joint sparse recovery method for compressed sensing with structured dictionary mismatches. *IEEE TSP*, 62(19):4997–5008, 2014.

O. Teke, A. C. Gurbuz, and O. Arikan. Perturbed orthogonal matching pursuit. *IEEE TSP*, 61(24):6220–6231, 2013.

J. Zhang, L. Chen, P. T. Boufounos, and Y. Gu. On the theoretical analysis of cross validation in compressive sensing. In *ICASSP*, pp. 3370–3374, 2014.

H. Zhu, G. Leus, and G. Giannakis. Sparsity-cognizant total least-squares for perturbed compressive sampling. *IEEE Trans. Signal Process.*, 59(11), 2011.

Xiaofan Zhu and Michael Rabbat. Graph spectral compressed sensing for sensor networks. In *ICASSP*, pp. 2865–2868, 2012.

Xiuming Zou, Lei Feng, and Huaijiang Sun. Compressive sensing of multichannel EEG signals based on graph fourier transform and cosparsity. *Neural Processing Letters*, 51:1227–1236, 2020.

A Appendix

Please see the supplemental material for proofs of the theorems, and many additional empirical results.