Graph Signal Processing for Scene Representation and Analysis

Tian, Dong; Mansour, Hassan; Cohen, Robert A.; Vetro, Anthony

TR2016-064 May 25, 2016

Abstract

Graph signal processing (GSP) is an emerging field that provides a new family of tools for analyzing signals that could be modeled on vertices connected by edges. In this paper, we describe two examples of how GSP is being applied for scene representation and analysis, where the scene is either captured as video sequences or point clouds. In the first example, we show that novel graph constructions can be used to robustly segment moving foreground objects from the background of video sequences with ego-motions. In the second example, we employ a graph-based transform to efficiently code attributes associated with the point clouds. We demonstrate with the two examples the potential benefits of using GSP tools for scene representation and analysis.

Graph Signal Processing Workshop (GSP)

Graph Signal Processing for Scene Representation and Analysis

Dong Tian, Hassan Mansour, Robert Cohen, Anthony Vetro
Mitsubishi Electronic Research Labs (MERL)
201 Broadway, Cambridge, MA 02139, USA
{tian; mansour; cohen; avetro}@merl.com

***Abstract*—Graph signal processing (GSP) is an emerging field that provides a new family of tools for analyzing signals that could be modeled on vertices connected by edges. In this paper, we describe two examples of how GSP is being applied for scene representation and analysis, where the scene is either captured as video sequences or point clouds. In the first example, we show that novel graph constructions can be used to robustly segment moving foreground objects from the background of video sequences with ego-motions. In the second example, we employ a graph-based transform to efficiently code attributes associated with the point clouds. We demonstrate with the two examples the potential benefits of using GSP tools for scene representation and analysis.**

I. Introduction

We address applications for deriving scene semantics that help analyze the motion content of videos and finding compact representations of visual point cloud data using graph-based signal processing. Motion is an important cue in video segmentation since rigid objects often exhibit similar motions within their parts. Point cloud data captured by scanning the 3-D world is another type of data that is used to represent the scene structure. There is a strong need to represent the point cloud in an efficient way so as to facilitate its storage, transmission and analysis.

In general graph signal processing [1], an undirected graph $G = (V, E)$ consists of a collection of nodes $V = \{1, 2, ..., N\}$ connected by a set of links $E = \{(i, j, w_{ij})\}$, $i, j \in V$ where (i, j, w_{ij}) denotes the link between nodes i and j having weights w_{ij}. The adjacency matrix $\mathbf{W}$ of the graph is an $N \times N$ matrix with weights w_{ij} as its entries, and the degree d_i of a node i is the sum of link weights connected to node i.

In this paper, we describe our recent work on applying graph signal processing tools for two applications in efficient scene representation and analysis: motion segmentation and point cloud coding. For image processing applications, a pixel may be treated as a node in a graph; and for point clouds, each point can be a node.

II. Motion Segmentation based on Graph Constraints

A. Overview

Sparse subspace clustering (SSC) has recently been used as a robust algorithm for motion segmentation using graph spectral clustering techniques [2], where feature point trajectories are extracted from several video frames. The limitation is its reliance on computing trajectories across multiple images. Liu et al. [3] proposed a graph spectral clustering based on feature descriptors of superpixels assuming that an object appears in multiple images. In another related approach [4], a "hypergraph" is built rather than a traditional graph where the hypergraph is constructed based on similarities defined on higher order tuples rather than pair of nodes.

A common limitation observed with the above methods is their inability to cope well with complex motions, especially with strong perspective effects appearing in the scene. In this section, we enforce geometric constraints by defining a graph structure where the signal on the graph is the set of motion vectors (MVs) computed between every two frames from a video encoder. Therefore, our approach does not rely on detecting or tracking feature points or extracting pixel trajectories over more than two frames.

B. Geometric-guided Graph Approaches

Unlike geometric vanishing points that are derived from the lines appearing on objects in a scene, *motion vanishing points* (MVP) are incurred at the intersections between motion vectors.

Just as points from a rigid object with translational motion share the same MVP in 3-D world, their corresponding motion vectors in the image would share the same MVP in the 2-D image plane. Based on this observation, the distances between motion vanishing points can be used to distinguish moving objects and to group together pixels from one object even when parts of the object have motions in different directions due to perspective effects. Therefore, we construct a graph as shown in the left figure in Fig. 1, where the graph weights are assigned as perspective distances based on MVP analysis as shown in the right figure in Fig. 1.

The principle eigenvectors of the constructed graph, i.e. those eigenvectors among the first K eigenvectors $\{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_K\}$ after removing the eigenvectors corresponding to zero eigenvalues, could be used to cluster the pixels into k groups [5].

Since spectral clustering does not provide direct semantic meaning for each cluster, we further propose to utilize a label propagation method to assign semantic meaning to the

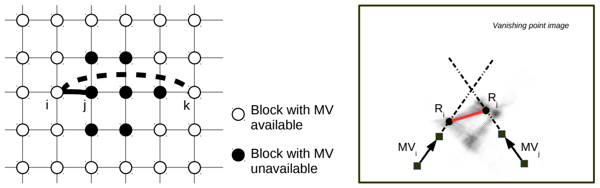

Fig. 1. Left: Graph structure; Right: Representation point of a MV and distance between a MV pair.

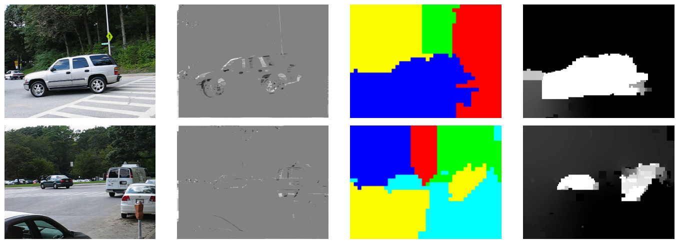

Fig. 2. **From left to right:** Original image, result of using only PCA, result of spectral clustering with the graph only clustering and the final result assisted with label propagation.

segmented clusters by using PCA results as initial labels. Some test results on Hopkins 155 Dataset are shown in Fig. 2.

III. Graph Processing for Point Cloud Compression

A. Background

3-D point clouds have become a practical way to represent spatial data in many applications, such as virtual reality, mobile mapping, and scanning of historical artifacts. In particular, a point cloud comprises a set of coordinates or meshes, representing the point locations and some attributes attached to each point. Earlier work from the computer graphics community compressed or reduced the size of point clouds primarily by leveraging the connectivity of structured point clouds. Such approaches achieved compression by reducing the number of vertices in triangular or polygonal meshes by, for example, fitting surfaces or splines to the meshes [6].

In this section, we present a way to improve coding efficiency when graph transforms are used for lossy compression on blocks partitioned from a large or sparse point cloud.

B. Graph Transform with K- Nearest Neighbors

The graph transform for coding attributes in point clouds has been studied in [7], which is limited in that the points are aligned to grid positions, and only the points that are one unit apart in any dimension are identified as being connected when constructing the graph. This approach does not permit multiple attributes to be co-located on one grid point, so resampling or attribute averaging of those points would be needed. Sampling the grid at a finer resolution would avoid this problem, but it could lead to many disjoint sub-graphs in each block, which can impact coding efficiency.

Here, we construct a graph with edges being more flexibly placed between points, allowing the points being located on quantized or fractional positions. Instead of limiting neighbors

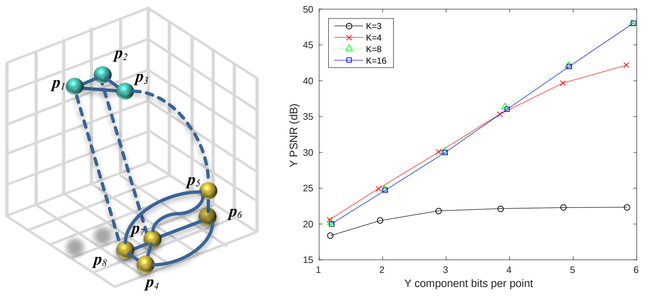

Fig. 3. Left: Illustration of process going from separate sub-graphs to one graph by using K-NN with $K = 3$. Right: Performance of K-nearest-neighbor graph transform for different values of K with $pr = 1.0$

to being one unit apart, we connect each point to its K nearest neighbors and then prune any duplicated edges. In this way, the K-nearest-neighbors method (K-NN) is able to incorporate more regular connections within the graph, and thus leads to a more appropriate graph transform.

On the left of Fig. 3, an example K-NN graph structure is shown with $K = 3$, where the K-NN approach will make new connections (P_1, P_8), (P_2, P_7) and (P_3, P_5), as compared to the graph constructed using [7]. However, the K-NN graph is not guaranteed to encompass all points of a block in one sub-graph. Additional processes could be developed and subsequently applied to determine how to connect multiple disjointed sub-graphs. It is worth noting that in the K-NN graph after pruning, there may exist some points which are connected to more than K points. For example, P_7 in Fig. 3 has 5 associated edges.

Finally, the weight of the graph edge is a function of distance, e.g. $e^{-\frac{d}{2\sigma^2}}$, where d is a measured distance between the connected points and σ denotes the variation in statistics of the points.

Compression performance on *Statue_Klimt_PointCloud* for varying K is shown in the right figure of Fig. 3. As expected, severely limiting the connectivity of points to at most $K = 3$ neighbors hinders performance. $K = 4$ performs well for lower rates, and the best performance is achieved with $K \geq 8$. Increasing K well above 8 or 16 yields the same performance.

IV. Conclusion and Other Work

We demonstrated two applications of graph signal processing for visual scene representation and analysis. First, in order to handle challenging motion segmentation for video sequences with ego-motion, a geometric guided graph method is proposed that utilizes graph spectral clustering. Second, for point cloud processing, a K-NN graph transform is investigated to efficiently code the attribute associated with each point.

We also use the GSP tools for other related applications, e.g. motion analysis for car driving sequences, object tracking over time, graph resampling for point cloud, etc. We expect GSP tools to be useful for many applications in scene representation and analysis, as graph could be viewed as a newer way to represent constraints among the pixels/points in a scene.

References

[1] D. Shuman, S. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *Signal Processing Magazine, IEEE*, vol. 30, no. 3, pp. 83–98, 2013.

[2] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 11, pp. 2765–2781, 2013.

[3] Y. Liu, J. Liu, Z. Li, J. Tang, and H. Lu, "Weakly-supervised dual clustering for image semantic segmentation," in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. IEEE, 2013, pp. 2075–2082.

[4] P. Ochs and T. Brox, "Higher order motion models and spectral clustering," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 614–621.

[5] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*. MIT Press, 2001, pp. 849–856.

[6] A. Maglo, G. Lavoué, F. Dupont, and C. Hudelot, "3D mesh compression: Survey, comparisons, and emerging trends," *ACM Comput. Surv.*, vol. 47, no. 3, pp. 44:1–44:41, Feb. 2015.

[7] C. Zhang, D. Florencio, and C. Loop, "Point cloud attribute compression with graph transform," in *2014 IEEE International Conference on Image Processing (ICIP)*, Oct. 2014.