Effective graph sampling of a nonlinear image transform

Mark de Lancey[1] and Inger Fabris-Rotelli[2][0000−0002−2192−4873]

[1] Department of Statistics, University of Pretoria, South Africa
mark.stephen.del@gmail.com
[2] Department of Statistics, University of Pretoria, South Africa
inger.fabris-rotelli@up.ac.za

Abstract. The Discrete Pulse Transform (DPT) makes use of LULU smoothing to decompose a signal into block pulses. The most recent and effective implementation of the DPT is an algorithm called the Roadmaker's Pavage, which uses a graph-based algorithm that produces a hierarchical tree of pulses as its final output. This algorithm has been shown to have important applications in artificial intelligence and pattern recognition. Even though the Roadmaker's Pavage is an efficient implementation, the theoretical structure of the DPT results in a slow, deterministic algorithm. This paper examines the use of the spectral domain of graphs and designing graph filter banks to downsample the Roadmaker's Pavage algorithm. We investigate the extent to which this speeds up the algorithm and allows parallel processing. Converting graph signals to the spectral domain can also be a costly overhead, and so methods of estimation for filter banks are examined, as well as the design of a good filter bank that may be reused without needing recalculation.

Keywords: Discrete Pulse Transform · Graph Sampling · multiscale

1 Introduction

The Discrete Pulse Transform (DPT) decomposes a signal into block pulses using LULU smoothers in such a way that the signal can be reconstructed fully [1]. The LULU smoothers L_k and U_k are applied recursively from $k = 1$ to K to obtain the DPT, the sequential decomposition of a signal f (such as an image) into scale levels $D_1(f), D_2(f), D_3(f), ..., D_K(f)$ such that the sum of these gives the original signal $f = \sum_{k=1}^{K} D_k(f)$. Each scale level consists of block pulses (connected components) of size k. This in turn has been used to detect features in signals and extract textures from images by partial reconstruction of the pulses. Feature and texture extraction has important applications in artificial intelligence, pattern recognition and computer vision [4]. Applying LULU operators directly on a signal using first principles until the signal is fully decomposed results in an operation of $\mathcal{O}(N^3)$ complexity [2]. To create a more feasible implementation, a graph based algorithm known as the Roadmaker's algorithm was developed, which reduced the computational complexity to $\mathcal{O}(N)$ [3]. The main shortfall of the Roadmaker's algorithm is that its storage of block pulses requires a hierarchical tree containing sparse matrices at each node, which makes extraction of the pulses (and therefore reconstruction of the image) slow as well as storage requirements of the data structure relatively high. An improvement on this comes in the form of the Roadmaker's Pavage algorithm [5]. This algorithm is also a graph based implementation, however the final data structure produced does not require sparse matrix storage at each node. The decomposition stage of the Roadmaker's Pavage is still $\mathcal{O}(N)$ and does not give an improvement on decomposition time, its advantage comes in the form of an improved resulting data structure that requires significantly less storage as well as a faster reconstruction and access time.

It should be noted that although the implementation algorithms of the DPT have reduced to linear complexity, the algorithms are still computationally slow. This is due to the algorithms needing to be processed in series, as well as the comparisons and transformations of data structures required at each step. Hence the true computational times needed is somewhat masked by the Big-$\mathcal{O}$ complexity. There is still a need to reduce to computational time, particularly for real-time application. The fact that these newer implementations are in the form of graph structures provides several advantages, especially considering the recent advances in theory and application of graph sampling and interpolation methods.

The research conducted here makes use of such recent developments in graph sampling and graph spectral theory to improve the running time and memory requirements of the Roadmaker's Pavage algorithm by using an approximation of the algorithm and its output. In addition to this, any advantages that come with graph spectral analysis is now also introduced to the algorithm. The primary mechanism behind these improvements comes from the use of graph spectral filters and graph sampling methods.

The paper begins with an overview of the suggested algorithm and its components in section 2, and then demonstrates application of the algorithm in section 3, before concluding.

2 Methodology

The method used to improve computational speed of the Roadmaker's Pavage algorithm involves the use of a graph spectral filter bank. The reason behind filtering the signals is in order to band limit the signal. Band limited graph signals can be interpolated with perfect reconstruction after sampling under certain conditions [9]. The graph filter bank used makes use of two filters (low-pass and high-pass) to split the original image into its low and high frequency components. The high and low frequency signals are then passed on to each pipeline of the bank at which point each signal is operated on independently of the other. The remaining operations after filtering include:

1. Downsampling the filtered signals
2. Performing the DPT decomposition using Roadmaker's Pavage on the filtered signals

3. Full or partial reconstruction of the pulses stored within the tree structure
4. Upsampling the reconstructed signal
5. Filtering again the upsampled signals
6. Summing the signals and multiplying them with a constant to obtain the final output signal

Hence the algorithm results in two smaller trees containing low frequency and high frequency pulses. After reconstruction of the desired pulses the signals produced are then upsampled, filtered again and combined together to obtain the final output signal. Each filter and pipeline operates completely independent from the other, and extraction of pulses from either tree is independent from each other. Because of this, the new algorithm has become what is known as embarrassingly parallel, that is, the job of parallel processing the filter bank and signals is trivial as the only time the two independent pipelines need to communicate with each other is at the very end where the two output signals are added together. The Roadmaker's Pavage wedged between filter banks is shown in Figure 1. The individual components of the filter bank and algorithm are explained next.

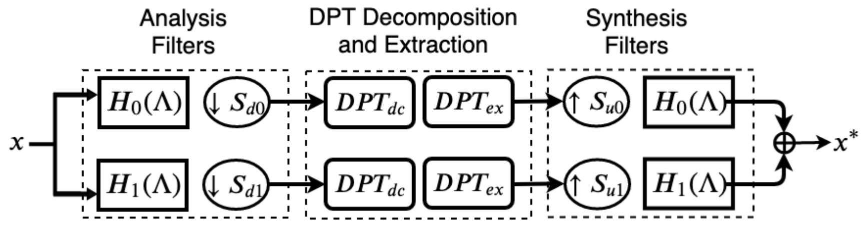

Fig. 1. A diagram of a filter bank with the Roadmaker's Pavage algorithm wedged in between each pipeline

2.1 Roadmaker's Pavage decomposition and extraction DPT_{dc} and DPT_{ex}

The Roadmaker's Pavage is a graph-based algorithm that results in a tree that contains the information required to extract the same pulses as defined by the DPT. The algorithm starts by imposing the image on a rectangular grid graph, known in this context as the Working Graph. Through a series of edge contractions, comparisons and clusterings, this Working Graph is eventually transformed into a tree. An example of the pulses extracted from a small 2×2 image, as well as the data structures built by the Roadmaker's Pavage is shown in Figure 2.

The Working Graph that is initialized is an example of a rectangular grid graph, which is also a bipartite graph. Even though the final product of the

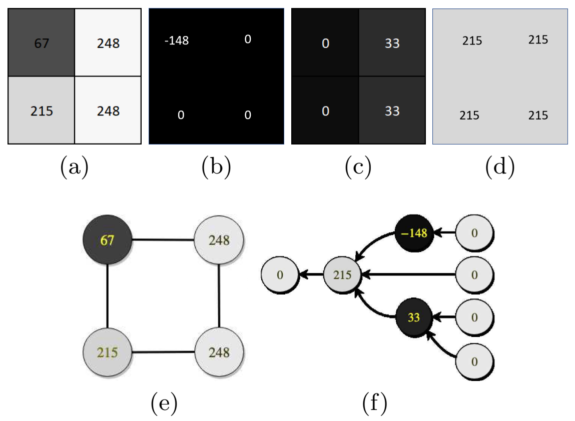

Fig. 2. Examples of a DPT decomposition using the Roadmaker's Pavage algorithm. Showing (a) the original 2×2 image. (b)-(d) The separate pulses of the image such that $(b) + (c) + (d) = (a)$. (e) The image imposed on the working graph. (f) The final pulse graph, a directed rooted tree, which is the result of completion of the Roadmaker's Pavage. The pulses shown in $(b) - (d)$ can be extracted from this tree.

Roadmaker's Pavage is a tree, the extracted pulses need to be reshaped into an image with the same dimensions as the original in order to be meaningful. The reshaped output signal is not contained in a graph, but its properties and relations still behave as if it was imposed a new grid graph with the same dimensions as the original working graph. Thus, filtering, sampling and interpolation is based on this Working Graph structure. The original signal is on a graph of this type and is filtered and sampled from accordingly, while the output signal is upsampled and filtered as if it were a signal on a new grid graph. For these reasons the Roadmaker's Pavage decomposition, as well as extraction of the desired pulses, is inserted into the middle of the filter bank.

2.2 Graph Spectral Filters $H_0(\Lambda)$ and $H_1(\Lambda)$

$H_0(\Lambda)$ and $H_1(\Lambda)$ are low and high pass filters respectively. They first convert a graph signal, $\boldsymbol{x} \in \mathbb{R}^{\mathbb{N}}$ with each x_i defined on vertex i, into its graph spectral domain using the Graph Fourier Transform (GFT). The signal is then converted back to the vertex domain using the Inverse Graph Fourier Transform (IGFT). The Graph Fourier Transform can be obtained from the eigendecomposition of a graph shift operator [14, 12, 13, 11, 9]. An example of a graph shift operator is the adjacency matrix, $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ where $\boldsymbol{A_{ij}} = \boldsymbol{A_{ji}} = 1$ if $i \neq j$ and v_i is adjacent to v_j otherwise $\boldsymbol{A_{ij}} = \boldsymbol{0}$ (for undirected, unweighted graphs).

An alternative shift operator derived from $\boldsymbol{A}$ is the Laplacian Matrix, $\boldsymbol{L}$ [14, 12, 13, 11, 10]. This is the difference of a graphs degree matrix $\boldsymbol{D}$ (whose diagonal entries gives the number of edges protruding from the corresponding node) and its adjacency matrix. Hence the Laplacian is defined by $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$. Finally, the most important shift operator to be used in this paper is the symmetric normalized Laplacian, $\boldsymbol{\mathcal{L}}$ [12, 13]. The symmetric normalized Laplacian is given by $\boldsymbol{\mathcal{L}} = \boldsymbol{D^{-\frac{1}{2}}LD^{-\frac{1}{2}}} = \boldsymbol{I} - \boldsymbol{D^{-\frac{1}{2}}AD^{-\frac{1}{2}}}$, with:

$$\boldsymbol{\mathcal{L}_{ij}} = \begin{cases} 1 & \text{if } i = j \text{ and } deg(v_i) \neq 0 \\ -\frac{1}{\sqrt{deg(v_i)deg(v_j)}} & \text{if } i \neq j \text{ and } v_i \text{ is adjacent to } v_j \\ 0 & \text{otherwise.} \end{cases}$$

As the name implies, this version of the Laplacian is always symmetric, hence its eigenvector basis is real and orthogonal [11, 13]. Some other important properties of the symmetric normalized Laplacian in the spectral domain include the fact that its eigenvalues range from 0 to 2 [11, 13] and since the graph used in the Roadmaker's Pavage is bipartite (described in section 2.3) its eigenvalues are also symmetric around 1. Now that $\boldsymbol{\mathcal{L}}$ has been defined, its eigendecomposition is given by

$$\boldsymbol{\mathcal{L}} = \boldsymbol{U\Lambda U^T},$$

where $\boldsymbol{\Lambda}$ is a diagonal matrix containing the eigenvalues $\{\lambda_1, \lambda_2, ..., \lambda_N\}$ of $\boldsymbol{\mathcal{L}}$ and the columns of $\boldsymbol{U}$ contains the eigenvectors of $\boldsymbol{\mathcal{L}}$. This eigenbasis is orthonormal, hence $\boldsymbol{U^{-1}} = \boldsymbol{U^T}$.

Definition 1. *The* ***Graph Fourier Transform*** *is the projection of a graphs signal in its vertex domain to the graphs spectral domain. For a diagonalizable graph shift operator $\boldsymbol{\mathcal{L}} = \boldsymbol{U\Lambda U^T}$ on graph G, the Graph Fourier Transform of graph signal $x \in \mathbb{R}^N$ is given by:*

$$\boldsymbol{\hat{x}} = \boldsymbol{U^T x}$$

Alternatively it may be defined by its evaluation at each eigenvalue:

$$\forall \lambda_k,\ \boldsymbol{\hat{x}}_k = (\boldsymbol{x}, v_k^T) = \sum_{i=1}^{N} \boldsymbol{x}_i \boldsymbol{u_{k,i}} = \boldsymbol{x^T u_k}$$

where $\boldsymbol{u_k}$ is the eigenvector associated with the k^{th} eigenvalue of $\boldsymbol{\mathcal{L}}$, which is also the k^{th} column of $\boldsymbol{U}$ [12, 11, 14].

The GFT is invertible, as $\boldsymbol{U\hat{x}} = \boldsymbol{UU^{-1}x} = \boldsymbol{x}$. With the GFT defined, graph spectral filters can be constructed.

Definition 2. *A* ***Graph Spectral Filter*** *is a function that projects separately on each of the eigenspaces of $\boldsymbol{\mathcal{L}}$ depending on the value of the respective λ. Mathematically, a kernel $\lambda \to h(\lambda)$, defines a graph filter $\boldsymbol{H}$ such that $H = \boldsymbol{U}h(\boldsymbol{\Lambda})\boldsymbol{U^T}$.*

Hence a filtered signal is defined such that $\boldsymbol{x_h} = \boldsymbol{Hx}$. Several kernels are suitable to define a filter. The simplest is to use an indicator function that cuts off certain frequencies completely, such as $h(\lambda) = 1$ if $\lambda < w$, else $h(\lambda) = 0$, where w is the desired cut-off frequency. The filter used here is a graph quadrature mirror filter (graph-QMF) so that:

1. H_0 and H_1 are used in both the analysis and synthesis bank of each respective pipeline, as opposed to different filters used at each end,
2. $H_0 = H_1(2 - \lambda)$. Recall that the eigenvectors of $\mathcal{L}$ are symmetric around 1 and range from 0 to 2, and
3. $H_0^2(\lambda) + H_1^2(\lambda) = c^2$

where $1/c^2$ is a constant that scales the final output signal [13]. For a simple indicator kernel where $w = 1$, $c^2 = 2$.

Finally, of note is the fact that exact calculation of these graph filters requires calculation of the full eigenspectrum as well as the dense eigenvector matrix $\boldsymbol{U}$ of size N^2 which can require more memory space than available for large N. For this reason, Chebyshev polynomials are used to approximate the response of $x_h = \boldsymbol{U}h(\boldsymbol{\Lambda})\boldsymbol{U^T x}$. This is because $h(\lambda)$ can be approximated as a Kth order polynomial $\sum_{k=0}^{K-1} a_k \lambda^k$ [10].

2.3 Upsampling and downsampling operators $\downarrow S$ and $\uparrow S$

The operators S_{d0}, S_{d1} and S_{u0},S_{u1} are used to downsample and upsample signals, respectively. Downsampling is done by simply taking only the $n < N$ vertices and corresponding signals in the graph that are chosen so that $\mathbb{R}^N \to \mathbb{R}^n$. Upsampling increases the dimension of a sampled signal to the original graph dimensions, imposing the signal on the original vertices and then fills any missing values with zeroes. The initial graph used in the Roadmaker's Pavage algorithm is a grid graph, which is also a bipartite graph. A bipartite graph is a graph that can be divided into two disjoint and independent sets $\mathcal{V}_{Bottom}$ and $\mathcal{V}_{Top}$ such that every edge connects a vertex in $\mathcal{V}_{Bottom}$ to one in $\mathcal{V}_{Top}$, while the vertices within a set are not connected to each other. Sampling a bipartite graph signal according to these disjoint bipartite sets is a standard procedure when sampling is required, as the sets will cover a large area of the graph with approximately equal spacing between both sampled and excluded vertices. This makes interpolation of an upsampled signal more accurate as each empty excluded node will have several neighbours used to interpolate its value [13, 12].

There is however, a major caveat when using bipartite sampling for the Roadmaker's Pavage. By definition, each bipartite set has no connections within itself but the Roadmaker's Pavage requires a connected graph in order to perform the required transform and comparisons. To remedy this, an adjusted graph is used for sampling. First, vertex indices to be sampled are chosen in a bipartite manner as usual. Additional diagonal edges are then used to join nodes together. Finally the two sets are separated from each other as if the graph is still bipartite, but now each set is connected. Thus the Roadmaker's Pavage algorithm can proceed on these sampled graphs. An example of this procedure performed on a small graph with 4 nodes is shown in Figure 3.

3 Application

In this section a comparison is shown between using the original Roadmaker's Pavage algorithm on an example image (of Chelsea the cat), and the proposed

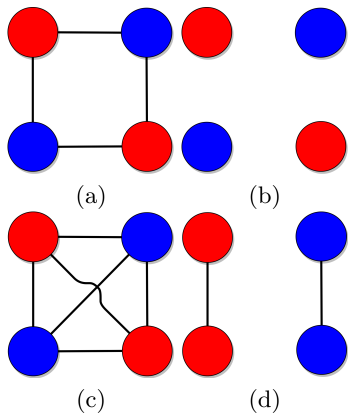

Fig. 3. An example of bipartite graph sampling with added edges The graphs are coloured according to their disjoint bipartite sets: $\mathcal{V}_{Bottom}$ in blue and $\mathcal{V}_{Top}$ in red. Shown: (a) original toy graph, (b) disconnected graph after bipartite sampling, (c) graph with added diagonal edges, (d) two connected graphs after sampling using indices as if bipartite.

Roadmaker's Pavage used in conjunction with a filter bank. The original image was decomposed into pulses using both methods. In the case of the sampled and filtered algorithm, it was found that the entire high frequency pipeline could be discarded for the cost of a small decrease in mean-squared-error and extraction accuracy of smaller features. This meant that only one line of filters and samplers was required in this instance as well as only a single tree data structure.

The image was then reconstructed both fully and partially in both cases. Partial reconstruction included extraction of only the smallest textures in the image, extraction of a mid-range of pulse sizes giving a smoothed image and extraction of the largest pulses giving the large features present in the image. The full and partial reconstruction of the image can be seen in Figure 4 while the equivalent reconstructions can be seen in Figure 5.

The original image has 135 300 pixels. The root-mean-squared-error between the fully reconstructed original image and the interpolated fully reconstructed image from the filtered Roadmaker's Pavage was 1.47. The original algorithm has an RMSE of zero as it perfectly reconstructs the original image. The error introduced by the filtered algorithm can be justified by the noticeable decrease in computational time and storage requirements. The original algorithm required 87 CPU seconds to decompose, approximately 5.5 CPU seconds to reconstruct the signals and needed a tree with 170 777 nodes to store the information. The filtered algorithm needed only 27.5 CPU seconds to decompose the signal, approximately 2.5 CPU seconds to reconstruct and a tree with 89 234 nodes for storage. A summary of these findings is given in Table 1

The size of the sampled graphs is approximately half the size of the original. The size and distribution of pulses extracted depends on the signal and size of

Table 1. Summary of application findings.

Measurement	**Original Algorithm**	**Proposed Algorithm**
Pulse Graph size (number of nodes)	170777	89234
Decomposition Time (in CPU seconds)	87.0	27.5
Reconstruction Time (in CPU seconds)	5.5	2.5
RMSE against original image	0.00	1.47

the graph used. No precise method to find equivalent pulses between the full Roadmaker's Pavage output and the filtered and sampled Roadmaker's Pavage output. Future work will develop adaptive selection of the pulses using shape measurements of pulses.

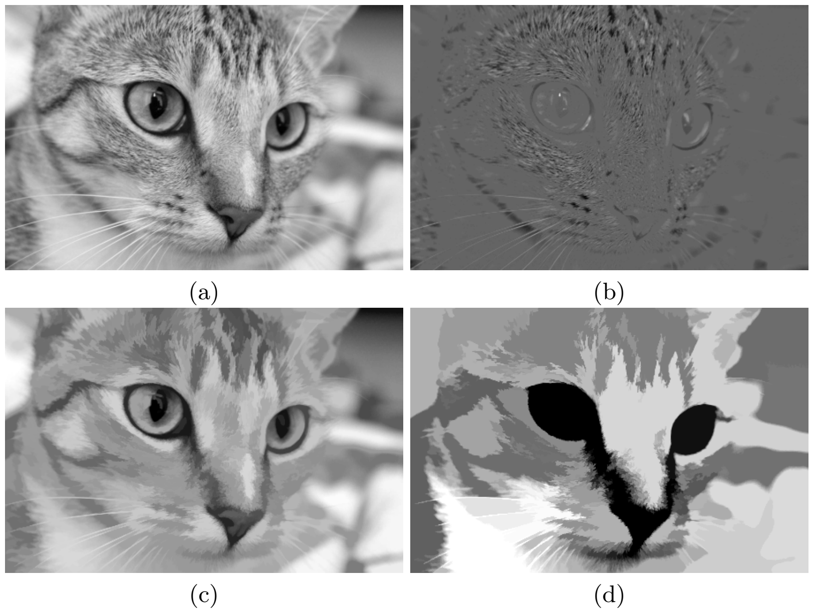

Fig. 4. An image of Chelsea decomposed and reconstructed using the Roadmaker's Pavage. Shown in: (a) original image, (b) smallest textures extracted, (c) smooth image extracted, (d) largest features extracted.

4 Conclusion

This paper has shown that an approximation of the Discrete Pulse Transform can be obtained using the Roadmaker's Pavage algorithm in conjunction with

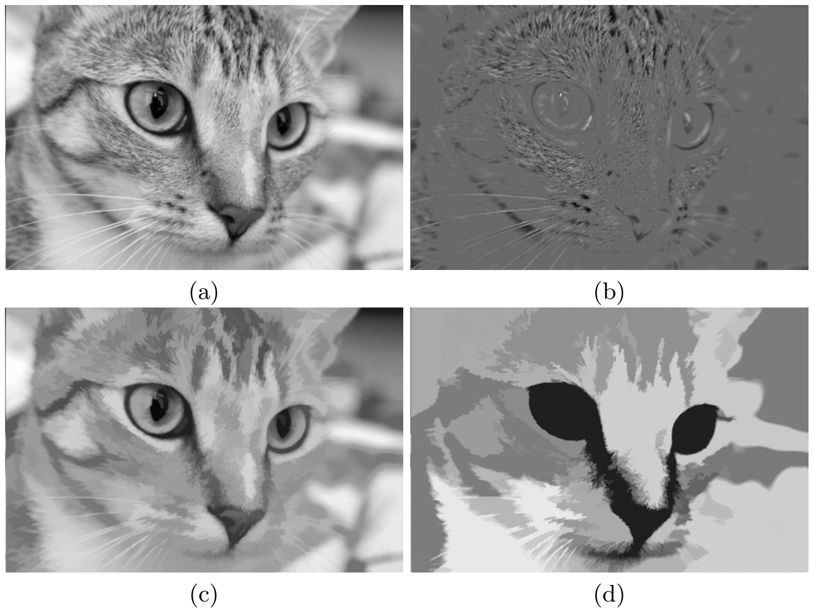

Fig. 5. An image of Chelsea decomposed and reconstructed using the Roadmaker's Pavage via a graph bank that has been upsampled, filtered and interpolated after the pulses were reconstructed. Shown in: (a) interpolation of all pulses reconstructed, (b) interpolation of smallest textures, (c) interpolation of smooth image, (d) interpolation of largest features.

graph spectral filtering and sampling. A minor loss of accuracy comes with the benefits of noticeably shorter computational time and storage requirements. Depending on the level of precision required and the size of the features needed, the high pass filters can be discarded if not necessary for the task. Even if the high frequencies are still needed, this approximate algorithm can now be calculated with two independent channels and thus can be parallel processed. Simulations are to be conducted in the future on sets of images to find the distribution and consistency of these findings. The partially reconstructed images here were obtained through trial and error to find rough equivalents between the two methods. For these reasons no mean square-error or similarity comparison was yet made to compare partial reconstruction. These will be compared in simulations in the future using trial methods such as approximating the equivalent pulse sizes needed by checking the relative pulse distributions of the original Roadmaker's Pavage and the new filtered Roadmaker's Pavage, and shape measures. This sampling approach for the DPT should be further tested for accuracy in areas the DPT has been applied such as segmentation [6], feature detection [7] and its effectiveness when considering leakage [8].

References

1. Fabris-Rotelli, I., Anguelov, R.: LULU operators and Discrete Pulse Transform for multidimensional arrays. IEEE Transactions on Image Processing **19**(11), 3012-3023 (2010)
2. Laurie, D.P.: The Roadmaker's Algorithm for the Discrete Pulse Transform. IEEE Transactions on Image Processing **20**(2), 361-371 (2010)
3. Laurie, D.P., Rohwer, C.H.: Fast implementation of the Discrete Pulse Transform. In: Psihoyios, G., Simos, T., Tsitouras, C. (eds.) International Conference of Numerical Analysis and Applied Mathematics 2006, pp 484-487. Weinheim, Germany (2006)
4. Laurie, D.P., Rohwer, C.H.: The Discrete Pulse Transform, SIAM Journal on Mathematical Analysis (SIMA) **38**(3), (2007).
5. Stoltz, G. G.: Roadmaker's pavage, pulse reformation framework and image segmentation in the Discrete Pulse Transform. University of Pretoria (2014)
6. Fabris-Rotelli, I., Greeff, J.F.: The application of the iterated conditional modes to feature vectors of the Discrete Pulse Transform of images. In: De Waal, A. (eds.) Proceedings of the 23nd Annual Symposium of the Pattern Recognition Association of South Africa 2012, pp 149 - 156 (2012). ISBN: 978-0-620-54601-0
7. Fabris-Rotelli, I.: The Discrete Pulse Transform for images with entropy-based feature detection. In: Robinson, P., Nel, A. (eds.) Proceedings of the 22nd Annual Symposium of the Pattern Recognition Association of South Africa 2011, pp 43 - 48 (2011). ISBN: 978-0-620-51914-4
8. Fabris-Rotelli, I., Stoltz, G.: On the leakage problem with the Discrete Pulse Transform decomposition. In: De Waal, A. (eds.) Proceedings of the 23rd Annual Symposium of the Pattern Recognition Association of South Africa 2012, pp 179-186 (2012). ISBN: 978-0-620-54601-0
9. Siheng, C., Varma, R., Sandryhaila, A., Kovačević, J.: Discrete Signal Processing on Graphs: Sampling Theory. IEEE Transactions on Signal Processing **63**(24), 6510 - 6523 (2015)

10. Shuman, D.D., Vandergheynst, P., Frossard, P.: Chebyshev polynomial approximation for distributed signal processing. In: International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS) (2011). https://doi.org/10.1109/DCOSS.2011.5982158
11. Tremblay, N., Gonçalves, P., Borgnat, P.: Design of graph filters and filterbanks (2017)
12. Sakiyama, A., Watanabe, K., Tanaka, Y., Ortega, A.: Two-Channel Critically Sampled Graph Filter Banks With Spectral Domain Sampling. IEEE Transactions on Signal Processing **67** (6), 1447 - 1460 (2019)
13. Narang, S.K.,Ortega, A.: Perfect Reconstruction Two-Channel Wavelet Filter Banks for Graph Structured Data. IEEE Transactions on Signal Processing **60**(6) , 2786 - 2799 (2012)
14. Cheung, G., Magli, E., Tanaka, Y., Ng, M.K.: Graph Spectral Image Processing. Proceedings of the IEEE **106**(5), 907 - 930 (2018)