Prediction of Centroid Pixel Values in Image Triangulations Using a Graph Neural Network

Luka Lukač
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor, Slovenia
E-mail: luka.lukac@um.si

Andrej Nerat
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor, Slovenia
E-mail: andrej.nerat@um.si

Damjan Strnad
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor, Slovenia
E-mail: damjan.strnad@um.si

Filip Hácha
Department of Computer Science and Engineering
University of West Bohemia
Plzeň, Czech Republic
E-mail: hachaf@kiv.zcu.cz

Borut Žalik
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor, Slovenia
E-mail: borut.zalik@um.si

***Abstract*—Image triangulation is a simple and abstract representation of an image. Important image structures are represented with triangles that are scattered across the image in an unstructured manner. However, quite often, when dealing with an image triangulation, the user's object of interest is the original image. Various interpolation methods have been used in order to predict the original pixel values inside the triangulation simplices. Although yielding accurate results in some cases, their results can be significantly inaccurate when dealing with high-frequency details in simplices of the image triangulation, or if the triangulation simplices have a highly irregular structure. In this paper, a new interpolation method based on a graph neural network is proposed. The experimental results on the popular dataset DIV2K showed that the proposed method, in most cases, produces smaller prediction errors than the existing interpolation methods, such as Barycentric Coordinates or Inverse-Distance Weighting.**

Keywords-image processing; Delaunay triangulation; machine learning; Graph Neural Network; interpolation.

I. Introduction

Image compression methods often use prediction methods to achieve better compression ratios. Neighbouring pixels in a real-life image are usually highly correlated [1]. Therefore, the current (unknown) pixel value can be predicted fairly accurately from previously-encoded pixels in the close neighbourhood with a non-complex texture [2]. Such a prediction approach assumes that pixels are encoded in a pre-determined order, which represents a serious limitation in the case of non-structured data, such as irregularly sampled pixels. In the case of an image triangulation, the key pixels can be scattered across the raster space without a specified order, which is why conventional methods based on image convolution cannot be applied directly to predict unknown values of pixels that are not a part of the triangulation.

In another part of computer science, Geographic Information Systems (GIS), data sampling locations are usually also distributed irregularly across the observed area [3]. Quite often, the object of our interest are locations with no available data [4]. To enable performing environmental analyses at such locations, many interpolation methods have been developed in the past [5]–[8]. In terms of images, interpolation methods have been used mostly for tasks such as super-resolution [9] and steganography [10].

In contrast to many machine learning methods, Graph Neural Networks (GNNs) represent a method that operates on a graph domain instead of in the Euclidean space [11]. In the past, GNNs were used for a variety of tasks (e.g., materials science [12][13], recommendation systems [14], and natural phenomena forecasting [15]). There are applications of GNNs in the field of image processing as well, including tasks such as super-resolution [16][17], structural image classification [18], and image clustering [19]. However, despite the fact that graph representation is well-suited and used widely for the prediction of values in locations with unknown data, none of the existing methods deal with pixel values' prediction in image triangulations.

This paper presents a novel method using a GNN to perform the task of centroid pixel values' prediction in greyscale image triangulations. The remainder of the paper is structured in the following way: in Section II, the proposed prediction method is described in detail, Section III summarises and discusses the results of the experiment, while Section IV concludes the paper.

II. METHOD

The proposed method consists of three major parts:

- detection of the key pixels, where pixels are detected that carry important information about an image,
- graph construction, where the detected key pixels are transformed into a graph with the Delaunay triangulation [20]–[22] to which additional, centroid pixels' nodes are added,
- centroid pixel value predictions, where a GNN is utilised to predict the value of a centroid pixel inside the corresponding triangulation simplex.

In the continuation of this section, each part is described in detail.

A. Detection of Key Pixels

Let I be a greyscale image embedded into a raster space with x columns and y rows. In the first step of the method, the key pixels $\mathcal{P}^k = \{p_i^k\} \subseteq I$ are detected using one of the established methods for image feature detection. Key pixels can, in principle, represent various important image features. However, in our case, the most beneficial features in an image are pixels with maximum gradients (i.e., edges and corners) as they carry the most important information about the image structure. Therefore, the most suitable methods for feature selection are edge detectors and corner detectors (e.g., Scale-Invariant Feature Transform (SIFT) [23] or Features from Accelerated Segment Test (FAST) [24]). Those methods are slightly adapted in our case, enabling a user to determine the rate r of all pixels in I (with maximum gradients) that shall be considered key pixels. An example of a greyscale image and its corresponding set of detected key pixels are displayed in Figure 1.

(a)

(b)

Figure 1. Detection of the key pixels: (a) A greyscale image I [25], (b) Detected key pixels $\mathcal{P}^k$ ($r = 0.05$).

B. Graph Construction

In the next step, a graph $\mathcal{G} = (V, E)$ is constructed, where $V = \{v_i\}$ denotes its vertices, while $E = \{e_{i,j}\}$ represents its edges. Firstly, the method constructs $\mathcal{G}$ with a Delaunay triangulation of $\mathcal{P}^k$ (an example is shown in Figure 2). After that, the centroid pixels $\mathcal{P}^c = \{p_i^c\}$ of the triangulation simplices are calculated, added to the graph, and marked as vertices that shall be predicted with a GNN. Three additional, secondary edges are formed from each p_i^c to all three vertices of the simplex in which the currently observed centroid is located. Lastly, the edge weights are calculated according to (1).

$$w_{i,j} = \begin{cases} 1 & d(v_i, v_j) = 0 \\ \frac{1}{d(v_i, v_j)} & \text{otherwise} \end{cases} \quad (1)$$

where $w_{i,j}$ denotes the weight between the i-th and j-th vertices, while d represents the Euclidean distance between two vertices.

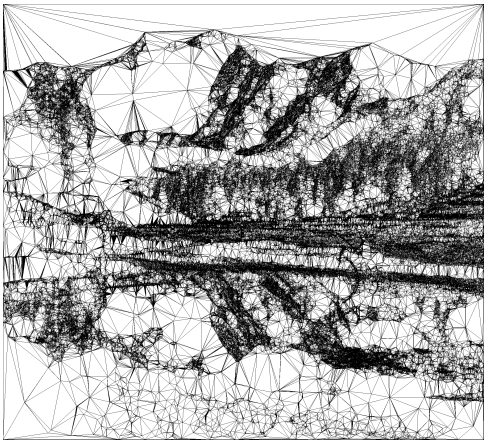

Figure 2. Delaunay triangulation of $\mathcal{P}^k$.

C. Centroid Pixel Value Predictions

After the construction of $\mathcal{G}$, a GNN is used to predict the values of $\mathcal{P}^c$. The GNN consists of two main parts: a graph convolution sequence and a linear sequence (as shown in Figure 3).

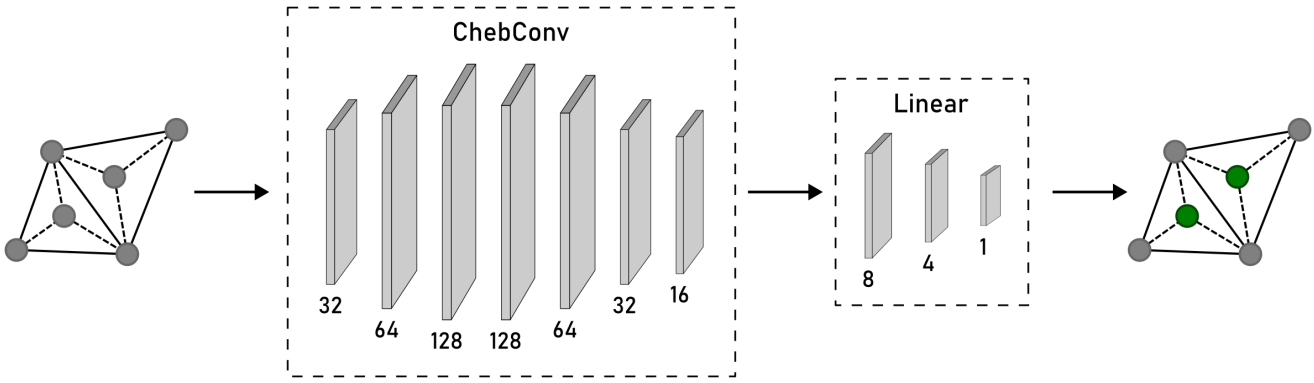

Figure 3. Design of the GNN: the embedding of the graph $\mathcal{G}$ is passed through ChebConv and Linear layers. A graph embedding, containing predictions of centroid pixel values (marked with green), represents the output of the GNN.

As an input, the GNN receives $\mathcal{G}$, that is, afterwards, passed through 7 Chebyshev graph convolutional (ChebConv) layers [26] with the value $K = 3$. Rectified Linear Unit (ReLU) activation function is applied after each layer. The number of ChebConv layers indicates that the result of the convolution is composed of vertices, which are 7 edge connections apart from the current vertex at most. The initial value of p_i^c is set to the average pixel value of its corresponding triangulation simplex. The embedding of $\mathcal{G}$ is constructed from the initial centroid pixel values of $\mathcal{P}^c$. The first ChebConv layer transforms the initial embedding into a graph representation with 32 channels, while the following ChebConv layers transform the intermediate representation into representations with 64, 128, 128, 64, 32, and 16 channels, respectively. In the second part, the values of $\mathcal{P}^c$ are predicted using 3 fully-connected layers with 8, 4, and 1 output, sequentially. The output of the final fully-connected layer (after applying the ReLU activation function) represents the graph embedding that includes the predictions of the centroid pixel values.

III. Results

The results of the method are presented and briefly discussed in this section. One of the most popular image datasets, DIV2K [25], was used for the training of the GNN. Among 1,000 photos, 800 were selected for the training of the GNN, 100 for the validation, and 100 for testing purposes. The image dataset was augmented, in order to reduce the overfitting effect of the neural network. During the data augmentation, the detection of key pixels in each image was performed in a way where their rate varied from 2% to 10% of the total pixel count in an image (effectively producing 9 different key pixel sets from one image). Min-max normalisation of the pixel values and graph attributes was performed, in order to rescale the features to the interval [0,1]. The GNN was implemented with the framework PyG (PyTorch Geometric) [27] and trained on NVIDIA GeForce RTX 3080 Graphics Processing Unit (GPU).

The hyperparameters of the GNN were tuned with a random search, and are summarised in Table I. However, there were some limitations that had to be considered while tuning the hyperparameters. As DIV2K contains large images, their graph representations are very memory-demanding. Consequently, the batch size for training had to be set to 1 in order to prevent running out of memory on the GPU. Furthermore, as the training phase was significantly time-demanding due to large graph representations, the number of epochs was limited to 10. After each mini-batch, the training loss was calculated on $\mathcal{P}^c$.

TABLE I. TRAINING HYPERPARAMETERS OF THE GNN.

Number of epochs	10
Learning rate	0.001
Batch size	1
Optimisation algorithm	Adam [28]
Loss function	Mean Squared Error (MSE)

The results of the GNN were compared with two popular methods used for interpolation: Barycentric Coordinates (BC) and Inverse-Distance Weighting (IDW). Root-Mean-Square Error (RMSE) was used to evaluate the error of the predictions on the test set (within the pixel range [0,255]). The experimental results of the three methods are presented in Table II.

TABLE II. AVERAGE PREDICTION ERROR WITH DIFFERENT INTERPOLATION METHODS.

Method	**RMSE**
BC	22.54
IDW	23.23
GNN	**20.26**

The comparison between BC, IDW, and the GNN revealed that, on average, our method outperformed BC by 10.11% and IDW by 12.79%. The latter indicates that our method significantly improves the prediction accuracy. Furthermore, the results of the experiment showed that, among 900 test images, the GNN outperformed BC and IDW in 839 cases, meaning that our method's prediction produced the best result in 93.22% of the test samples. Graphically, the results of the tests are shown in Figure 4.

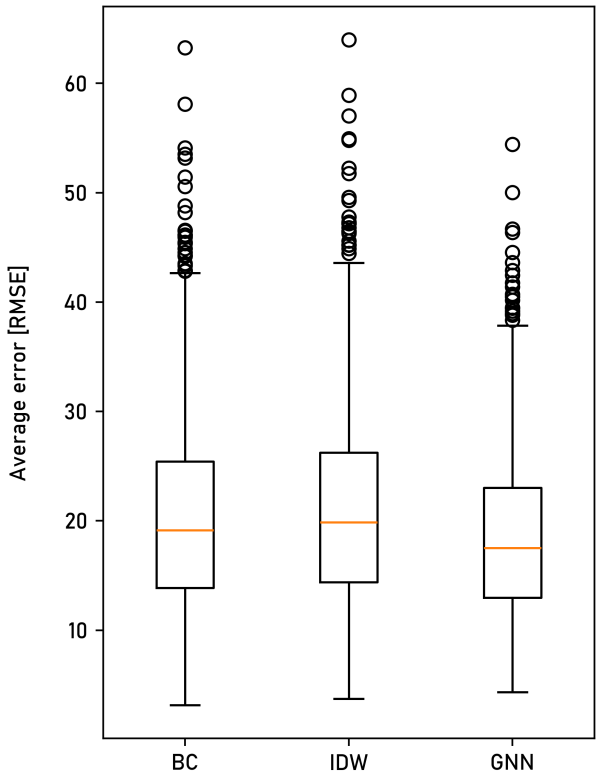

Figure 4. Average errors of the tested interpolation methods: BC, IDW, and the GNN.

The GNN-based prediction achieved the best result in terms of median error and InterQuartile Range (IQR). The upper 1.5*IQR whisker of the GNN also lies significantly lower than the BC and IDW counterparts, which indicates that the image samples with larger error are less common when predicting pixels with the GNN. The only case where both BC and IDW performed better than the GNN is the lower 1.5*IQR whisker, indicating that, in some cases, conventional interpolation methods can outperform machine learning methods.

IV. Conclusion and Future Work

A new method for the prediction of centroid pixel values in image triangulations with a GNN is introduced in this paper. The key pixels that are detected with one of the established methods for important image features extraction are used to form a graph with a Delaunay triangulation. After that, the unknown pixel values, lying at the centroids of the triangulation simplices, are predicted using the GNN. The proposed GNN was trained on a large dataset of diverse greyscale images, which enhanced its versatility and efficiency. The GNN significantly outperformed the widely used conventional interpolation methods BC and IDW.

In the future, the proposed method could be integrated into algorithms for data compression that operate on unstructured data. In that case, it would probably be beneficial to perform prediction not only for centroid pixels but also for all non-key pixels in a raster space. A hierarchical GNN architecture [29] could be used for doing that. Such task, however, could be exceedingly demanding in terms of the computation power and the prediction performance. Another way in which we could continue our work is adapting our method to spatiotemporal domains such as video.

Acknowledgment

This research was funded by Slovenian Research and Innovation Agency under Research Project J2-4458 and Research Programme P2-0041, and the Czech Science Foundation under Research Project 23-04622L.

References

[1] L. Lukač, A. Jeromel, and L. Váša, "A short overview of prediction methods for image compression," in *Proceedings of the 32nd International Electrotechnical and Computer Science Conference*, (Portorož), pp. 145–148, IEEE Slovenia, September 2023.

[2] T. Dumas, A. Roumy, and C. Guillemot, "Context-adaptive neural network-based prediction for image compression," *IEEE Transactions on Image Processing*, vol. 29, pp. 679–693, 2020.

[3] J. Li and A. D. Heap, "Spatial interpolation methods applied in the environmental sciences: A review," *Environmental Modelling & Software*, vol. 53, pp. 173–189, 2014.

[4] L. Mitas and H. Mitasova, "Spatial interpolation," *Geographical Information Systems: Principles, Techniques, Management and Applications*, vol. 1, no. 2, pp. 481–492, 1999.

[5] D. Shepard, "A two-dimensional interpolation function for irregularly-spaced data," in *Proceedings of the 1968 23rd ACM National Conference*, ACM '68, (New York, NY, USA), p. 517–524, Association for Computing Machinery, 1968.

[6] R. Sibson, "A brief description of natural neighbor interpolation," *Interpreting Multivariate Data*, pp. 21–36, 1981.

[7] B. Yang, Q. Li, and W. Shi, "Constructing multi-resolution triangulated irregular network model for visualization," *Computers & Geosciences*, vol. 31, no. 1, pp. 77–86, 2005.

[8] A. Möbius, *Der barycentrische Calcul*. Johann Ambrosius Barth, 1827.

[9] W.-C. Siu and K.-W. Hung, "Review of image interpolation and super-resolution," in *Proceedings of The 2012 Asia Pacific Signal and Information Processing Association Annual Summit and Conference*, pp. 1–10, 2012.

[10] Y.-Q. Chen, W.-J. Sun, L.-Y. Li, C.-C. Chang, and X. Wang, "An efficient general data hiding scheme based on image interpolation," *Journal of Information Security and Applications*, vol. 54, p. 102584, 2020.

[11] J. Zhou *et al.*, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020.

[12] P. Reiser *et al.*, "Graph neural networks for materials science and chemistry," *Communications Materials*, vol. 3, no. 1, p. 93, 2022.

[13] V. Fung, J. Zhang, E. Juarez, and B. G. Sumpter, "Benchmarking graph neural networks for materials chemistry," *npj Computational Materials*, vol. 7, no. 1, p. 84, 2021.

[14] X. Li, L. Sun, M. Ling, and Y. Peng, "A survey of graph neural network based recommendation in social networks," *Neurocomputing*, vol. 549, p. 126441, 2023.

[15] F.-H. Zhang and Z.-G. Shao, "ST-GRF: Spatiotemporal graph neural networks for rainfall forecasting," *Digital Signal Processing*, vol. 136, p. 103989, 2023.

[16] S. Zhou, J. Zhang, W. Zuo, and C. C. Loy, "Cross-scale internal graph neural network for image super-resolution," *Advances in Neural Information Processing Systems*, vol. 33, pp. 3499–3509, 2020.

[17] T. Tarasiewicz, J. Nalepa, and M. Kawulok, "A graph neural network for multiple-image super-resolution," in *2021 IEEE International Conference on Image Processing (ICIP)*, pp. 1824–1828, 2021.

[18] A. Quek, Z. Wang, J. Zhang, and D. Feng, "Structural image classification with graph neural networks," in *2011 International Conference on Digital Image Computing: Techniques and Applications*, pp. 416–421, 2011.

[19] Y. Xing *et al.*, "Learning hierarchical graph neural networks for image clustering," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3467–3477, October 2021.

[20] B. Delaunay, "Sur la sphere vide," *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, vol. 7, no. 793-800, pp. 1–2, 1934.
[21] I. Kolingerová and B. Žalik, "Improvements to randomized incremental Delaunay insertion," *Computers & Graphics*, vol. 26, no. 3, pp. 477–490, 2002.
[22] B. Žalik, "An efficient sweep-line Delaunay triangulation algorithm," *Computer-Aided Design*, vol. 37, no. 10, pp. 1027–1038, 2005.
[23] D. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, pp. 1150–1157 vol.2, 1999.
[24] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Computer Vision – ECCV 2006* (A. Leonardis, H. Bischof, and A. Pinz, eds.), (Berlin, Heidelberg), pp. 430–443, Springer Berlin Heidelberg, 2006.
[25] E. Agustsson and R. Timofte, "NTIRE 2017 challenge on single image super-resolution: Dataset and study," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1122–1131, July 2017.
[26] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *arXiv*, 2017.
[27] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.
[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv*, 2014.
[29] S. Sobolevsky, "Hierarchical graph neural networks," *arXiv preprint arXiv:2105.03388*, 2021.