CLASSIFICATION OF VECTORIZATION METHODS IN THE RASTER2VECTOR MOBILE APPLICATION

Joldasov Shaxislam Batirovich
Tashkent University of Information Technologies named after Muhammad al-Khwarizmi

Abstract: *This article presents a classification and analysis of the vectorization methods implemented in the Raster2Vector mobile application. The study examines the interrelationships among the algorithms used in the raster-to-vector conversion process, including edge detection, image segmentation, contour extraction, and polygonal approximation. Furthermore, the image processing stages implemented using the OpenCV library and their impact on vectorization quality are discussed. The results demonstrate the effectiveness of the Raster2Vector mobile application in converting complex raster images into high-precision vector objects.*

Keywords: *Raster2Vector, vectorization, raster graphics, vector graphics, OpenCV, Canny Edge Detection, Gaussian Blur, segmentation, contour detection, polygonal approximation, mobile application.*

The rapid development of information technologies has increased the importance of graphic data processing and storage in various formats. In particular, the conversion of raster images into vector formats is widely applied in graphic design, engineering drawing, cartography, computer graphics, and digital archiving. Since raster images consist of pixels, they tend to lose quality when scaled or enlarged. In contrast, vector images are based on mathematical models and retain their quality regardless of scale.

The Raster2Vector mobile application has been developed for the Android platform to automatically convert raster images into vector representations. The application utilizes modern image-processing algorithms provided by the OpenCV library. The vectorization process consists of several stages, including image loading, color-space conversion, filtering, edge detection, segmentation, contour extraction, and transformation into vector objects.

The proper classification of vectorization methods is essential for evaluating application performance, identifying the strengths and limitations of different algorithms, and improving the system in future developments. Therefore, this article provides a functional and algorithmic analysis of the vectorization methods employed in the Raster2Vector mobile application.

Geometric transformation-based vectorization methods use geometric parameters to convert images into vector representations in the form of lines,

curves, or contours. This approach helps preserve the fundamental shapes of an image as geometric objects.

Contour-based vectorization methods focus on extracting and highlighting the contours of objects within an image, enabling the conversion of raster images into vector graphics. Particular attention is given to changes in pixel intensity that correspond to object boundaries.

Using the Sobel operator, image brightness gradients are extracted, and contours are identified through significant intensity variations. The Canny operator is employed for edge detection by applying noise filtering and edge-tracking algorithms. Meanwhile, active contour-based methods model object boundaries as dynamic curves that can adapt to the shape of image objects.

The Sobel operator is used to highlight image brightness gradients, thereby enabling contour detection. It is one of the most widely used edge detection methods based on calculating gradients along the X and Y axes. The Sobel operator measures changes in image intensity in both horizontal and vertical directions, making it possible to identify object boundaries where pixel intensity changes abruptly.

To calculate the horizontal gradient of an image, the following filter is applied: $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$, For the vertical gradient, the following filter is used: $G_x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$ These operators are applied to the image using convolution. The resulting operations produce gradients along the X and Y axes. The gradients are then combined to detect and extract image contours effectively.

Vectorization using Bézier curves enables the approximation of complex contours through mathematical curves, allowing shapes to be represented accurately in vector graphics. Image contours are approximated by a set of Bézier curves, which are smooth parametric curves controlled by multiple control points. This approach provides a compact and precise representation of object boundaries while maintaining the visual characteristics of the original image.

Texture and color-processing-based vectorization methods rely on the analysis of texture features and color distributions to transform images into vector representations. Texture-based image vectorization involves identifying and extracting repetitive patterns and structural regularities within an image. Such methods are particularly useful in applications where preserving texture characteristics is essential.[1]

Texture descriptors are used to identify and characterize texture features. This approach facilitates the extraction of properties such as texture orientation,

frequency, and intensity, which can subsequently be utilized for object classification and image analysis. By incorporating texture and color information, vectorization algorithms can achieve more detailed and semantically meaningful representations of image content.

The overall gradient magnitude G and gradient direction θ are then calculated as follows: $\sqrt{G_x^2 + G_y^2}$, $\theta = atan2(G_y, G_x)$, where $atan2(G_y, G_x)$ represents the gradient orientation angle. Assume that an image consists of pixels with varying intensity values. To calculate the image gradients and extract edges, the Sobel operators are applied. The horizontal and vertical gradients are first computed using the Sobel kernels, after which the gradient magnitude and direction are determined. Pixels with high gradient magnitudes correspond to regions of significant intensity change and are therefore identified as potential edges or contours. This process enables the detection and extraction of object boundaries within the image, which can subsequently be used for vectorization and shape representation.

The Canny operator is a more advanced edge detection method that consists of several stages: image smoothing, gradient computation, non-maximum suppression, and edge detection using threshold values. The operating principle begins with smoothing the image using a Gaussian filter to reduce noise, followed by the calculation of image gradients using the Sobel operator. Non-maximum suppression is then applied to retain only those pixels that correspond to local maxima along the gradient direction, enabling the separation of strong edges from weak ones.

The image smoothing process is expressed as: $I_{smooth} = I * G_\sigma$, where G_σ is the Gaussian filter and I represents the original image. The image gradients are calculated as: $G_x = \frac{\partial I_{smooth}}{\partial x}$, $G_y = \frac{\partial I_{smooth}}{\partial y}$. For this purpose, Sobel operators are used to determine the intensity variations along the X and Y axes. The resulting gradient values are then utilized to identify edge locations and orientations. By combining gradient analysis with non-maximum suppression and threshold-based edge tracking, the Canny operator provides accurate and reliable contour detection while minimizing the influence of image noise.[2]

The Canny operator is a more sophisticated edge detection method that consists of several stages: image smoothing, gradient computation, non-maximum suppression, and edge detection using threshold values. Its operating principle involves first smoothing the image with a Gaussian filter to reduce noise and then calculating image gradients using the Sobel operator. Non-maximum suppression is applied to retain only those pixels that correspond to the highest gradient values along the gradient direction, thereby distinguishing strong edges from weak ones.

Gradient computation: To determine changes along the X and Y axes, Sobel operators are applied. The gradient magnitude and direction are then calculated based on the horizontal and vertical derivatives.

Pixels that are not local maxima along the gradient direction are removed during the non-maximum suppression stage. Two threshold values, $T1$ and $T2$, are used to separate weak and strong edges. Pixels with gradient values greater than $T2$ are classified as strong edge pixels. Pixels with gradient values between $T1$ and $T2$ are considered weak edge pixels; however, they may still be preserved if they are connected to strong edges.[3]

Wavelet transformation makes it possible to analyze different levels of image detail, which is particularly important for extracting texture features. By decomposing the image into multiple frequency components, wavelet analysis provides both spatial and frequency information, enabling a more effective representation of texture patterns.

In this approach, the image is divided into small blocks, and histograms of gradients that characterize the texture are computed for each block. These histograms are then concatenated into a single feature vector, which serves as a compact representation of the image texture and can be used for further analysis, classification, or recognition tasks.

Vectorization methods based on color spaces utilize the color characteristics of an image, making it possible to preserve colors and shades that may be important for subsequent analysis. Converting RGB images into other color spaces such as HSV or LAB can enhance the distinction between object colors and improve segmentation quality. In addition, color histograms provide a representation of the image's color distribution in histogram form, which can be used for comparison, classification, and feature extraction tasks.[4]

In methods based on fractal theory, self-similar mathematical structures are used to vectorize images with complex and repetitive patterns. This approach is particularly useful for processing images containing natural or organic textures, such as trees, mountains, and other natural elements. Fractal-based techniques can effectively model irregular structures and preserve fine details during the vectorization process.

Machine learning-based vectorization methods utilize Convolutional Neural Networks (CNNs) to automatically extract important image features and convert them into vector representations. Pre-trained neural networks such as ResNet, VGG, Inception, and similar architectures are commonly employed for feature extraction. The input image passes through multiple convolutional layers, after which its extracted features are transformed into a vector representation suitable for further analysis. Contour-based vectorization methods focus on converting

raster images into vector forms by identifying and extracting object boundaries. Contours are lines that define the boundaries of objects within an image, allowing the image to be represented as geometric entities such as lines, curves, and polygons. These methods are widely used in various fields, including computer vision, image processing, and object recognition.[5]

Vectorization algorithms are particularly relevant for Geographic Information Systems (GIS), since the process of creating and updating digital maps is highly labor-intensive and is typically performed by experts. Automating the conversion of raster maps into vector formats significantly reduces processing time and improves the efficiency of spatial data management.

In the Raster2Vector mobile application, the vectorization problem is addressed using specialized software modules. The purpose of these modules is to process raster images of maps, extract useful information about the objects represented on the map, and store this information in specialized vector formats on storage devices or within databases. This class of software products is associated with cartographic data acquisition and processing. Since the primary analytical operations in Raster2Vector are performed using a vector data model, the application includes a wide range of functions for processing raster cartographic images and transforming them into structured vector representations suitable for further spatial analysis and visualization.

This stage addresses the problem of grouping, ordering, and determining the connectivity of points. The algorithms capable of solving this problem are quite diverse and, in general, can be reduced to contour tracing, convex hull construction, and clustering approaches.

The contour tracing method is one of the most widely used algorithms. It is relatively simple to implement, sufficiently fast, and highly efficient. As a result of raster-based processing, a list of boundary points can be obtained, as well as all other contour points arranged according to the contour traversal direction. These points are then used to form polygons that describe map regions. Due to the complexity of alternative approaches, they are not considered within the scope of this work. A correction stage is applied to eliminate redundant polygons. Typically, criteria such as the number of regions, their linear dimensions, and other geometric characteristics are used for this purpose. In the present study, the linear dimensions of the regions serve as the primary criterion.[8]

The formation of the vector format represents the final stage of the process, during which additional tasks are performed. These tasks include the computation of various statistical characteristics for both the entire image and its individual regions. The resulting vector data, together with other relevant information, are incorporated into a predefined data structure that can be stored on different storage

media and subsequently utilized by external applications. Such a structure is generally specified in advance according to the requirements of the target system.

All initial information about the map already exists and is stored in raster form. However, mathematical representation and analysis of such data are considerably more complex. As mentioned earlier, digital maps can be represented using either vector or raster formats. Before discussing vector representations, it is important to consider the structure and topology of raster data. Any image represented as a matrix whose elements correspond to color values can be considered a raster image.[6]

From an abstract perspective, a raster image may contain several types of regions. Examples of such regions are illustrated in Figure 1: convex regions (Figure 1a), non-convex regions (Figure 1b), and regions in the form of closed rings (Figure 1c). These different region types require specific processing approaches during the vectorization stage to ensure accurate representation of spatial relationships and geometric properties.

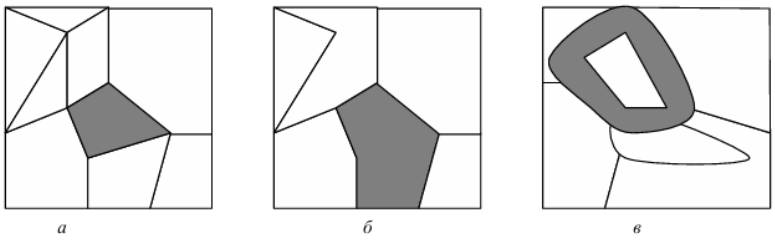

Figure 1. Representation of Regions with Different Colors in a Raster Image

Such diversity in shape classes generally increases the complexity of the vectorization problem. In this dissertation, the vectorization problem has been addressed for both convex and non-convex classes of shapes.

To adapt and modify the proposed algorithm, it is necessary to define only the contour traversal order, neighboring points, the selection of the starting point, and other related parameters. Once these elements are specified, the contour tracing procedure can be effectively applied to generate an accurate vector representation of the raster object boundaries. This approach simplifies the vectorization process while maintaining the geometric characteristics of the original regions and ensuring the correctness of the resulting vector model.

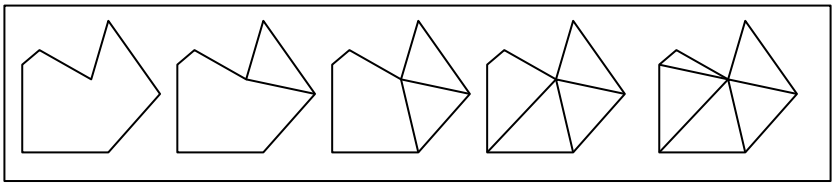

Figure 2. Illustration of the Triangulated Polygon Algorithm

There are numerous triangulated polygon algorithms, and the main drawback of this approach is the increase in the number of regions within the graph. As a result, the construction of the graph model requires greater memory resources and

higher computational costs. One possible solution to this problem is the application of algorithms that decompose a non-convex polygon into a set of convex polygons. Obviously, such algorithms are generally more complex and computationally demanding.[7]

The vectorization algorithm is presented in the form of general recommendations. This is due to the fact that the problem under consideration is often highly specific, and its solution depends on the characteristics of the particular application domain. Therefore, only a general framework for solving the required tasks is proposed. The suggested approach can be adapted and refined according to the structure of the input data, the geometric properties of the objects being processed, and the requirements of the target application.The results of this study indicate that the vectorization methods implemented in the Raster2Vector mobile application enable the accurate and efficient conversion of raster images into high-quality vector objects. Noise reduction through the Gaussian Blur filter, edge detection using the Canny algorithm, and the segmentation and contour extraction stages constitute the fundamental components of the vectorization process. Polygonal approximation algorithms simplify contours, reducing the size of vector data and making it more suitable for further processing.

Overall, the vectorization methods applied in the Raster2Vector mobile application provide a fast and effective solution for image processing on mobile devices. Future improvements may include the integration of Bézier curve generation, artificial intelligence, and machine learning techniques to further enhance vectorization accuracy and output quality.

REFERENCES:

1. Canny J. A Computational Approach to Edge Detection // IEEE Transactions on Pattern Analysis and Machine Intelligence. – 1986. – Vol. 8, No. 6. – P. 679–698.
2. Gonzalez R.C., Woods R.E. Digital Image Processing. – 4th ed. – New York: Pearson, 2018. – 1168 p.
3. Pratt W.K. Digital Image Processing: PIKS Scientific Inside. – 4th ed. – Hoboken: Wiley-Interscience, 2007. – 782 p.
4. Sonka M., Hlavac V., Boyle R. Image Processing, Analysis, and Machine Vision. – 4th ed. – Boston: Cengage Learning, 2014. – 920 p.
5. Szeliski R. Computer Vision: Algorithms and Applications. – 2nd ed. – Cham: Springer, 2022. – 979 p.

6. Beknazarova S.S., Joldasov Sh.B. Methods and algorithms for vectorization of raster images. J.: International engineering journal for research & development. VOL. 7 NO. ICMEI (2022). India-2022. - P.1-2

7. Beknazarova S.S., Joldasov Sh.B. Algorithm of generalization based on the minimum quadrates method. The American Journal of Engineering and Technology. ISSN 2689-0984, Volume 08 – 2026. -P.126-130

8. Beknazarova S.S., Joldasov Sh.B. Binary linear image vectorization algorithm. Journal of Computer Science Engineering and Information Technology Research (JCSEITR). ISSN(P): 2250-2416; ISSN(E): Applied Vol. 14, Issue 2, Dec 2025, -P.1-12

9.