ALGORITHMS AND ARCHITECTURE APPLIED IN THE RASTER2VECTOR SOFTWARE

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

Abstract: *This article examines the operating principles of the Raster2Vector software, focusing on the algorithms and software architecture employed in the vectorization process. The study discusses image processing techniques used to convert raster graphics into vector formats, including edge detection, contour tracing, and the generation of vector objects using geometric primitives. Furthermore, the interaction between software modules, data flow mechanisms, and architectural efficiency are analyzed. The results demonstrate the capability of Raster2Vector to perform high-precision vectorization and highlight its advantages in graphic data processing applications.*

Keywords: *Raster2Vector, vectorization, raster graphics, vector graphics, edge detection, image processing, algorithm, software architecture, Bézier curves, graphic data.*

Digital image processing relies heavily on both raster and vector graphics. Raster graphics consist of a collection of pixels and tend to lose quality when enlarged. In contrast, vector graphics are represented by geometric objects and mathematical formulas, allowing them to maintain quality regardless of scaling. Therefore, the automatic conversion of raster images into vector formats is an important task in graphic design, cartography, engineering drawing, and computer graphics.

The Raster2Vector software is specifically designed to address this challenge by analyzing raster images and generating vector shapes based on detected object boundaries. During operation, the software performs several stages, including image preprocessing, noise reduction, edge detection, contour tracing, and vector object generation. These stages involve the application of image segmentation, contour extraction, and curve approximation algorithms.

The architecture of Raster2Vector is based on a modular design, where each module performs a specific function. The input module receives raster images, the processing module analyzes and transforms image data, and the vectorization module generates vector elements based on the detected contours. The resulting data can be exported in vector formats such as SVG, DXF, or other supported standards. This approach enhances software efficiency while providing flexibility and compatibility with various graphic formats.

The overall architecture of the software is based on the MVVM (Model–View–ViewModel) architecture for Android applications, enabling raster images to be processed using the OpenCV library and exported into various vector formats. The MVVM (Model–View–ViewModel) pattern has been implemented to separate the user interface layer from the application logic[1].

Table 1.

Layers, Components, and Responsibilities Based on the MVVM Architecture

Layer	Component	Responsibility
VIEW	Fragments (14)	User interface implementation and event handling
VIEWMODEL	ConvertViewModel	Business logic, OpenCV image processing, and StateFlow management
MODEL	Data classes, Cache	Data structures and data storage management

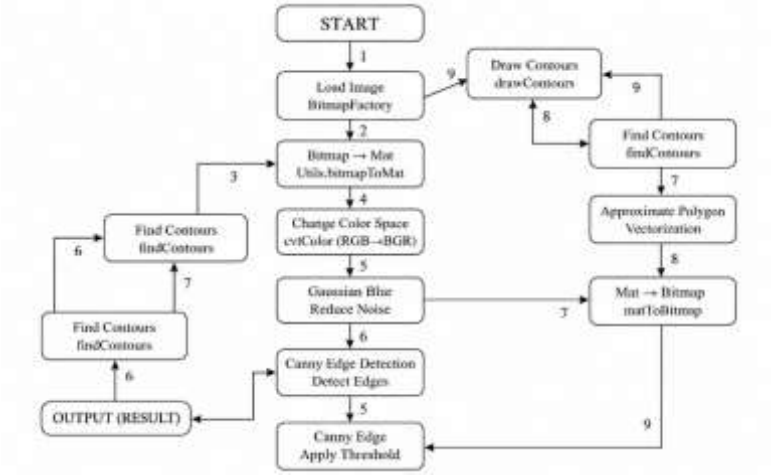

Figure 1. Architecture Diagram of the Raster2Vector Software

The database layer employs two different data storage mechanisms. One of them is SharedPreferences (Cache – Singleton Object). The Cache class is a lightweight wrapper built on top of SharedPreferences, designed to preserve user preferences and settings across application sessions[3].

Table 2.

Image Selection Module

Modules	Name
Fragment	SelectImageFragment.kt
Library	com.github.dhaval2404:imagepicker:2.1
Source	Camera or Gallery
Storage	getExternalFilesDir() + timestamp
Transfer	File path passed to the next fragment via Bundle

The user can either capture an image using the device camera or select an image from the gallery. The ImagePicker library simplifies the image selection process. The selected image is returned through the onActivityResult callback.

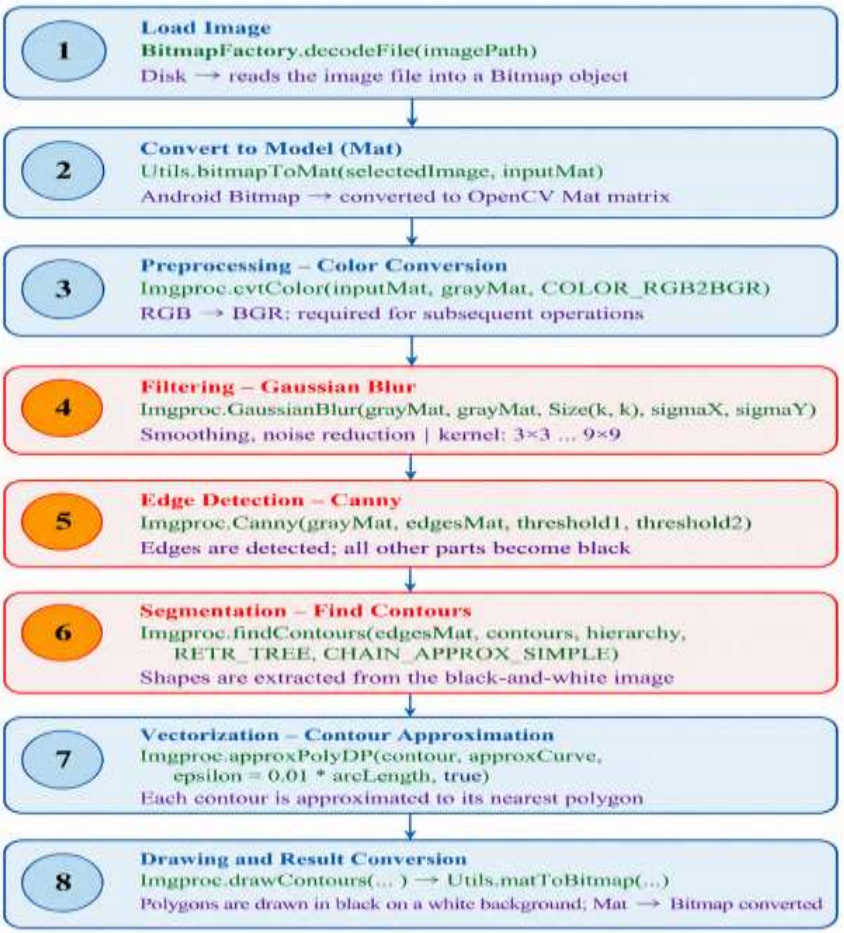

Figure 2. Stages of the Raster2Vector Mobile Application Workflow

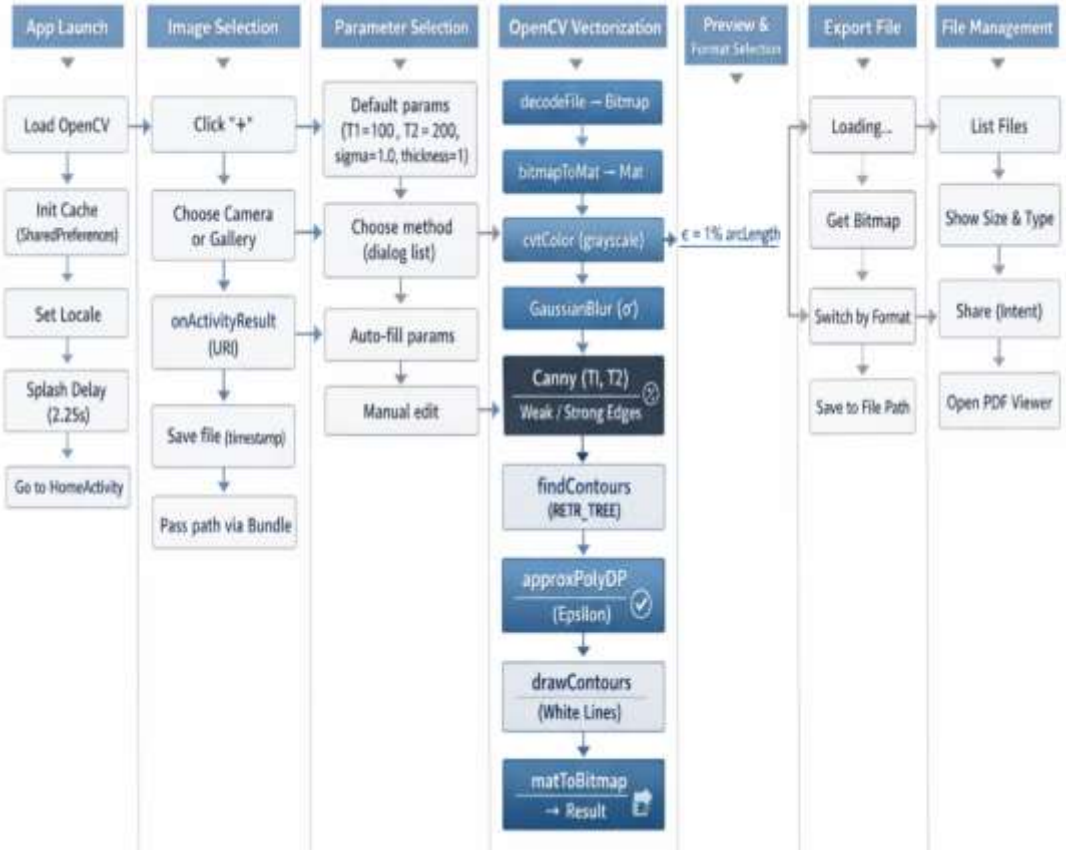

Figure 3. Stages of the Raster2Vector Mobile Application Workflow

Raster2Vector is a mobile application developed for the Android platform that converts raster images into high-precision vector formats. The software is based on the modern MVVM architecture, which enables a complete separation of the user interface from the business logic while ensuring system stability and maintainability. Within the application stack, the OpenCV library is utilized for digital image processing, SharedPreferences is employed for data caching, and external modules such as ImagePicker are integrated for image selection[2].

The operating principle of the application consists of a multi-stage processing pipeline. Initially, the image uploaded by the user is converted from a raster format into an OpenCV Mat matrix, and the color space is transformed from RGB to BGR. To reduce noise, a Gaussian Blur filter is applied, after which image edges are detected using the Canny edge detection algorithm. The core of the vectorization process is based on segmentation and polygonal contour approximation functions, through which raster pixels are transformed into vector representations composed of points, contours, and Bézier curve approximations[4].

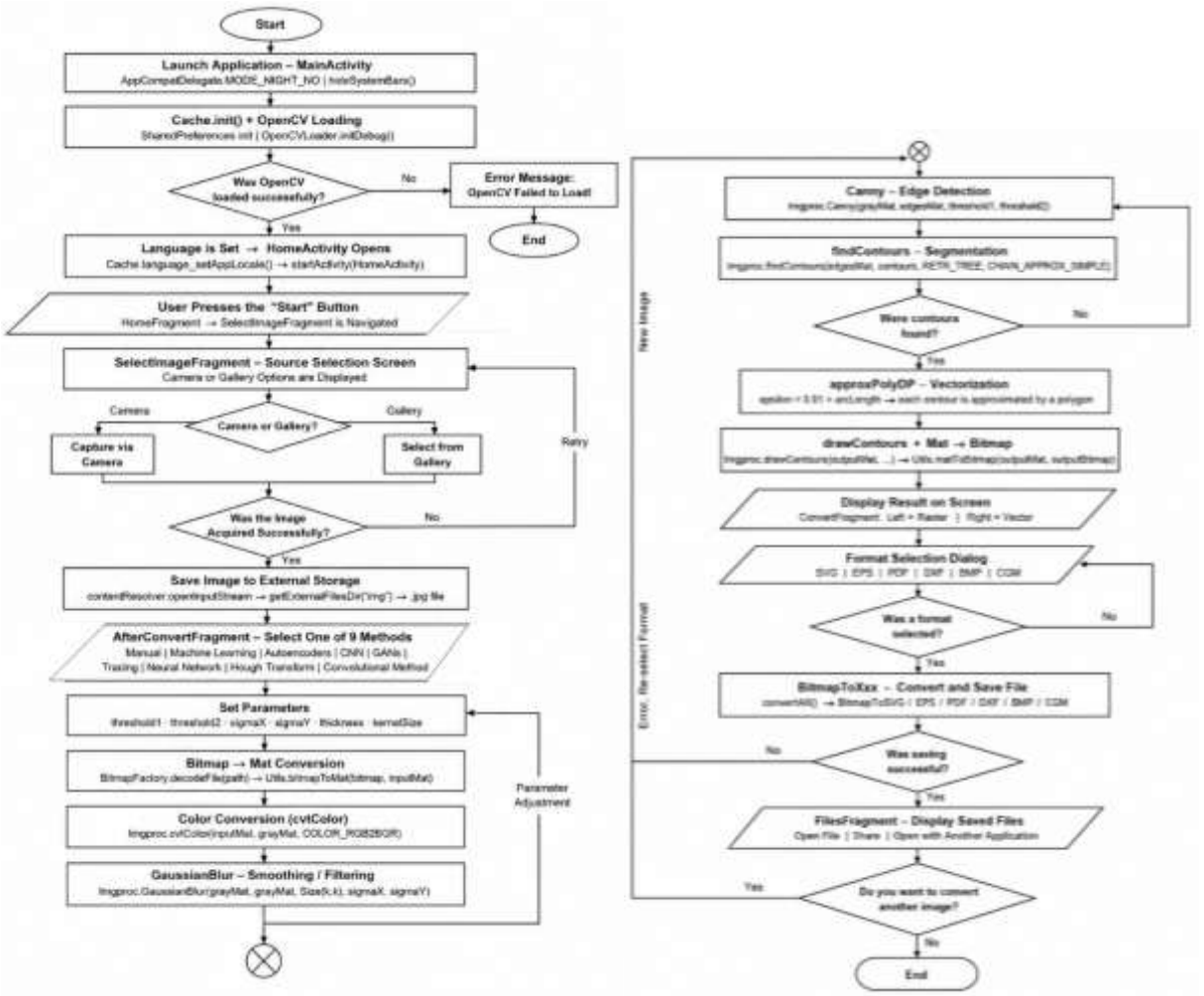

Figure 4. Algorithm of the Raster2Vector Mobile Application

The system provides a wide range of functional capabilities, allowing users to configure vectorization parameters either manually or automatically. The resulting vector images can be exported and stored in industry-standard formats such as SVG, PDF, EPS,

and DXF, as well as saved to external storage devices. The application is not only a collection of technical algorithms but also a comprehensive mobile working environment featuring intuitive navigation, a multilingual user interface, and an integrated file management system[3].

Raster2Vector software represents an effective solution for converting raster images into vector formats. The implemented algorithms for edge detection, segmentation, and geometric approximation enable accurate reconstruction of image objects and shapes. In addition, the modular architecture ensures flexibility, scalability, and ease of maintenance.

The analysis demonstrates that Raster2Vector is capable of producing high-quality vector representations even from complex graphical images, making it suitable for applications in graphic design, engineering projects, and digital archiving. Future improvements may include the integration of artificial intelligence and machine learning techniques to further enhance the accuracy and efficiency of the vectorization process. The analysis shows that Raster2Vector is capable of producing high-quality vector representations even from complex graphical images, making it suitable for applications in graphic design, engineering projects, and digital archiving. Future improvements may include the integration of artificial intelligence and machine learning techniques to further enhance the accuracy and efficiency of the vectorization process.

REFERENCES:

1. Туляганов Ш.Э. Разработка эффективных алгоритмов классификации объектов на основе комитетных решений: Автореф. дисс. … канд. техн. наук. – Ташкент, 2012.

2. 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

3. 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

4. Chen D., Daescu O. Space-Efficient Algorithms for Approximating Polygonal Curves in Two-Dimensional Space [Электронный ресурс]: поисковая база данных CiteSeer. – режим доступа: http://citeseer.ist.psu.edu/chen98spaceefficient.html, свободный.

5. Chen X., Gao W. Visual Pattern Recognition and Object Detection in Video Systems // IEEE Transactions on Circuits and Systems for Video Technology. – 2006. – Vol. 16, No. 9. – P. 1114-1123.