METHODOLOGY OF VECTORIZING BITMAP IMAGE

Saida Safibullayevna Beknazarova[1], Joldasov Shaxislam Batirovich[2]
[1]Dsc, professor, Tashkent University of Information Technologies named by Muhammad Al-Khwarizmi
[2]Researcher Tashkent University of Information Technologies named by Muhammad Al-Khwarizmi

https://doi.org/10.5281/zenodo.10511227

Abstract. *Vectorizing a bitmap image involves converting a raster image, composed of pixels, into a vector format consisting of paths or mathematical equations. This process allows for scalability and flexibility, as vector images can be resized without losing quality or becoming pixelated. In this article, we explore different techniques and approaches for vectorizing bitmap images, highlighting their benefits and limitations. Firstly, we delve into automated vectorization algorithms that use mathematical algorithms to trace the edges and colors of a bitmap image. These algorithms analyze the pixel values and attempt to recreate the image in a vector format. We discuss common techniques such as edge detection, color clustering, and region segmentation, evaluating their effectiveness in producing accurate vector representations. Next, we explore manual vectorization methods, which involve manually tracing or recreating the image using vector tools in graphic design software. We discuss the advantages of manual vectorization, such as greater control over detail and precision, as well as the additional time and effort required for this approach.*

Keywords: *vectorizing process, bitmap image, vectorization, CorelDRAW.*

Introduction

We also delve into the challenges and limitations associated with vectorizing bitmap images. These include the loss of fine details and textures during the conversion process, as well as difficulties in accurately reproducing complex shapes or gradients. We discuss strategies for overcoming these limitations, including combining automated and manual vectorization techniques. Furthermore, we examine the impact of vectorization on file size and compatibility. Vector files tend to be smaller than their raster counterparts, making them ideal for online distribution and storage. We also discuss different file formats, such as SVG and EPS, which are commonly used for vector images and their compatibility with various software and devices. Lastly, we discuss real-world applications and use cases for vectorized bitmap images. From graphic design and logo creation to architectural blueprints and technical illustrations, vectorization plays a crucial role in various industries. We highlight the benefits of vector images in terms of scalability, versatility, and ease of editing. In conclusion, vectorizing a bitmap image offers numerous advantages in terms of scalability, flexibility, and compatibility. Whether through automated algorithms or manual tracing, vectorization allows for the creation of high-quality images that can be easily resized and edited without loss of quality. However, it is important to consider the limitations and challenges associated with this process, as well as the specific use cases and applications where vectorization can be beneficial [1-3].

To vectorize a bitmap image, you can use various graphics programs and tools, including Adobe Illustrator, CorelDRAW, Inkscape, and many others. Here are the general steps for vectorizing a bitmap.

Import an image: Open the selected image editor and import your bitmap image.

Creating a new vector object: In the program, create a new vector object or layer that will contain the vectorized version of the image.

Vectorization Tools: Most image editors have vectorization tools, such as" Image Trace "in Adobe Illustrator or" Trace Bitmap " in Inkscape. Open this tool.

Configure Vectorization Parameters: Adjust vectorization parameters such as the level of detail, number of colors, and other parameters depending on your needs.

Methodology

Start the vectorization process: Start the vectorization process, which converts the bitmap image to a vector image based on the selected parameters.

Edit and Improve: After vectorizing the image, you may need to make adjustments and improvements to the vectorized image to achieve the desired result. This may include removing unnecessary details, editing polygons, and so on [4-6].

Save a vectorized image: When you are satisfied with the result, save the vectorized image in a suitable format, such as SVG or AI, so that it can be easily scaled and edited without losing quality.

This way, you can vectorize the bitmap image to get a vector version that can be scaled without losing quality and edited to suit your needs.

Bitmap vectorization is the process of converting a bitmap image (consisting of pixels) into a vector image, which is represented by geometric shapes such as lines, curves, and polygons. This allows the image to be scalable without losing quality and makes editing easier. There are several methods for vectorizing bitmaps:

Vectorization using software:

Adobe Illustrator: This program provides tools for vectorizing bitmaps, such as "Image Trace".

CorelDRAW: Similar to Adobe Illustrator, CorelDRAW provides tools for vectorization.

Inkscape: Free source software for editing vector images, which also includes tools for vectorization.

Vectorization using online services:

There are many online services that allow you to upload a bitmap image and get a vector version. For example, "Vector Magic" and "Autotracer.org".

Manual vectorization:

This method involves manually creating a vector image from scratch, using vector tools in image editors. This may require a lot of time and skill in drawing, but it allows you to get the best control over the process [6].

Machine learning methods:

There are machine learning techniques such as neural networks and computer vision algorithms that can automatically vectorize bitmaps. However, these methods are still under development and do not always produce perfect results.

Each of these methods has its own advantages and disadvantages, and the choice of method depends on the specific needs and skills of the operator. It is important to keep in mind that vectorization may require correction and manual work, especially when vectorizing complex bitmaps [7-9].

For automatic vectorization of bitmaps, neural networks and computer vision algorithms are used that convert pixel data from bitmaps to vector formats, such as SVG (Scalable Vector Graphics) or other vector formats. Below are some of the approaches and methods that can be used for this purpose:

Autoencoders: Autoencoders are neural networks that can learn a compact representation of an image. They can be trained to vectorize images by converting them to hidden space and then reconstructing them from it. This hidden representation can be used to create a vectorized version of the original bitmap image.

CNN (Convolutional Neural Networks): Convolutional neural networks are widely used in computer vision to extract features from bitmaps. You can train a CNN to extract vector features from bitmaps, and then use these features for vectorization.

Generative Adversarial Networks (GANs): GANs consist of a generator and discriminator that are trained together. The generator can create images that are similar to the input bitmaps, and this can be used to create vectorized versions.

Image processing techniques: There are also image processing algorithms such as contour tracing, Hough transform, anti-aliasing, and other techniques that can be used to vectorize bitmaps. These methods do not always require neural networks.

Discuss and results

Commercial and free programs: There are also specialized programs and online services that can automatically vectorize bitmaps. Some of them use a combination of different methods, including neural networks.

Convolutional neural networks for extracting features from bitmaps to use these features for vectorization.

Convolutional Neural Networks (CNNs) are widely used to extract features from bitmaps. Their specification, especially in image processing, makes them powerful tools for feature extraction and abstraction, which can be used to vectorize images, i.e. represent an image as a vector or a set of numeric features. Here's how it works:

Convolutional layers: CNNs typically include multiple convolutional layers that process the image using filters (kernels) to highlight various low-level features, such as faces, corners, and textures. These convolutional layers allow the neural network to automatically detect important features in the image.

Pooling layers: Convolutional layers are usually followed by pooling layers that reduce the dimension of features while preserving their important aspects. This reduces computational complexity and improves invariance to small changes in the image [11-13].

Fully connected layers: Convolutional and pooling layers are followed by fully connected layers that combine features from previous layers and create an output feature vector.

Vectorization: Output features from the last fully connected layers can be used to vectorize the image. This can be done by reducing the feature dimension and creating a vector that represents an abstract description of the image. This vector can be used for various tasks, such as classification, segmentation, search, and more.

Training: The entire CNN training process involves adjusting the weights of neural networks using marked-up data. This allows the model to learn which features are important for a particular task.

CNNs are widely used in computer vision for image-related tasks, such as object recognition, image classification, image segmentation, and more. Image vectorization using CNNs can be useful, for example, when searching for similar images, extracting textures, or creating abstract representations of images for higher-level tasks.

Generative Adversarial Networks for creating vectorized versions of an image that are similar to input bitmaps.

Generative Adversarial Networks (GANs) can be used to create vectorized versions of images that can resemble input bitmaps. The main idea of GAN is that two neural networks, a generator and a discriminator, compete with each other in the learning process.

The generator is responsible for creating images, while the discriminator tries to distinguish between authentic images and those generated by the generator. As you learn, the generator improves your skills by creating images that look more and more наauthentic.

To create vectorized versions of images using GANs, you can use a generator that can generate vectorized data, for example, SVG (Scalable Vector Graphics) or other vector formats instead of bitmaps. Training GANS to create vectorized images may require changes in network architecture and data processing approaches, but the basic concepts of GANS remain applicable.

Using GAN to create vectorized images may require additional processing and data transformation to convert bitmaps to vector formats and adapt the GAN architecture to work with vectors instead of pixels. This approach may require more complex architectures and specialized training methods to achieve high-quality results.

Methods: contour tracing, Hough transform, and anti-aliasing, which can be used to vectorize bitmaps.

Contour tracing for vectorizing bitmaps.

Contour tracing is the process of automatically or semi-automatically extracting the contours and borders of objects in bitmaps in order to create vector data. This process is often used to vectorize bitmaps to make the images scalable and workable in image editors or geographic information systems. Here are some ways to trace polygons:

Manual tracing: This method involves using a graphics editor such as Adobe Illustrator, CorelDRAW, or Inkscape to manually draw the contours of objects following the borders on the bitmap image. This method provides maximum control, but can be time-consuming for complex images.

Automatic Tracing: There are programs and tools that can automatically detect contours in bitmaps. Some of them use image processing techniques such as edge detection algorithms (such as Canny's algorithm) or contour detection algorithms to create vector contours. However, automatic tracing may require manual correction in case of complex images or poor quality of the source data.

Semi-automatic tracing: This method combines automatic and manual tracing. You can use tools that automatically detect polygons, and then edit them manually to improve accuracy and quality.

Machine learning: Some modern tools use machine learning algorithms, such as neural networks, to more accurately trace contours on bitmaps. These tools may be capable of more complex tasks, but they may require more computational resources and training data [13-14].

The choice of contour tracing method depends on the specific requirements and characteristics of the image. Manual tracing usually requires a lot of human input, but provides full control over the result. Automatic and semi-automatic tracing can be faster, but may require additional work to correct and improve the contours.

Neural networks can be used to trace contours on bitmaps, and this is one of the applications of machine learning in the field of computer vision. This process is often called contour segmentation or feature segmentation. Here are some steps that can be used for more accurate contour tracing using neural networks:

Data Preparation: First, you need to create a data set that contains bitmaps and marked-up polygons. Markup can be done manually, or using other methods, such as object detection algorithms [15-16].

Selecting a neural network architecture: Select the appropriate neural network architecture for the contour segmentation task. Convolutional neural networks (CNNs) are often used for processing bitmaps because of their ability to extract features from images.

Model Training: Train the selected neural network on the prepared data. Training involves feeding images to the network and correcting the weights of neurons so that they can accurately identify the contours of objects in the images.

Evaluation and tuning: After training the model, you need to evaluate its performance using a test dataset. If necessary, you can make adjustments to the network architecture or training parameters to achieve better results.

Inference: After successful training of the model, it can be used to segment contours in new images. The model will automatically highlight the contours of objects in images, which will provide more accurate contour tracing.

Conclusion

The use of neural networks for contour tracing usually gives good results, especially when working with complex and noisy images. However, it is important to choose the appropriate network architecture, configure the parameters correctly, and have enough marked-up data to successfully train the model.

Hough transform method, anti-aliasing for vectorizing bitmaps

The Hough transform method (or Hough-Red transform) is a mathematical method used to detect shapes in bitmaps.

The main idea of the Hough transform method is to translate image pixels into a parametric space, where straight lines or other shapes can be represented as points in this space. Then you can apply an algorithm to find groups of points that correspond to the same shape.

The Hough transform procedure for detecting straight lines usually looks like this:

For each pixel in the image that is part of a contour or edge, all possible parameters of a straight line passing through this pixel are calculated. For example, in the case of detecting straight lines, these parameters can be the distance from the origin to the straight line and the angle of inclination of the straight line.

Each set of parameters is represented in a parametric space. Points corresponding to straight lines are added to cells in this space.

The parametric space is analyzed to find peaks that indicate the possible presence of straight lines in the image.

Anti-aliasing (or filtering) can be applied to the image before applying the Hough method to reduce noise and improve the quality of shape detection. This may include various techniques, such as applying Gaussian filters, median filters, and other image processing techniques.

Using the Hough transform method with anti-aliasing can be useful for solving problems in computer vision, such as highlighting geometric shapes in images, detecting lines, circles, and other geometric structures.

REFERENCES

1. Schechner Y. Y., Narasimhan S. G., Nayar S. K., "Instant dehazing of images using polarization", IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 325-332, 2001.
2. Fattal R., “Single image dehazing”, International Conference on Computer Graphics and Interactive Techniques archive ACM SIGGRAPH, pp. 1-9, 2008.
3. Tan R. T. “Visibility in bad weather from a single image”, IEEE conference on Computer Vision and Pattern Recognition, pp. 1-8, 2008.
4. Kopf J., Neubert B., Chen B., Cohen M., Cohen-Or D., Deussen O., Uyttendaele M., Lischinski D., “Deep photo: Model-based photograph enhancement and viewing”, ACM Transactions on Graphics, Vol. 27, No. 5, pp. 116:1-116:10, 2008.
5. Fang S., Zhan J., Cao Y., Rao R. “Improved Single Image Dehazing
6. Using Segmentation”, IEEE International Conference on Image Processing (ICIP), 2010, pp. 3589-3592.
7. Матвеев Л. Т. Курс общей метеорологии. Физика атмосферы. – Л.: Гидрометеоиздат, 1984. – 752 с.
8. Beknazarova S., Mukhamadiyev A.Sh. Jaumitbayeva M.K.Processing color images, brightness and color conversion//International Conference on Information Science and Communications Technologies ICISCT 2019 Applications, Trends and Opportunities. Tashkent 2019
9. Beknazarova S., Mukhamadiyev A.Sh. Park Insu, Adbullayev S. The Mask Of Objects In Intellectual Irrigation Systems//International Conference on Information Science and Communications Technologies ICISCT 2020 Applications, Trends and Opportunities. Tashkent 2020.
10. Beknazarova S., Sadullaeva Sh., Abdurakhmanov K, Beknazarov K.. Nonlinear cross-systems of numerical simulation of diffusion processes//International Conference on Information Science and Communications Technologies ICISCT 2020 Applications, Trends and Opportunities. Tashkent 2020.
11. Korikov A.M. Correlation visual systems of robots / A.M. Korikov, V.I. Syryamkin, V.S. Titov. – Tomsk: Radio and Communications, 2000. – 264 p.
12. Klevalin V.A. Systems of technical vision in industrial robotics / V.A. Klevakin, A.Yu. Polivanov // Mechatronics, automation, control. - 2010. – No. 9. – pp. 26-36.
13. Goritov A.N. Selection of parametrically specified objects in a low–resolution image / A.N. Gorinov, S.I. Yakovchenko // Reports of TUSUR. – 2017. -Vol. 20, No. 2. – pp. 88-90.
14. Recommendation ITU-R BT.709-6. Parameter values for the HDTV standards for production and international programme exchange. – 2015. – P. 19.
15. Gonzalez R. Digital image processing / R. Gonzalez, R. Woods. – M.: Technosphere, 2005. – 1072 p.
16. Otsu N. A threshold selection method from gray-level histograms// IEEE Transactions on Systems, Man and Cybernetics. – 2009. – Vol. 9, № 1. – P. 62–66.