Multi-Stage Edge Detection for Generative Spatial Robotic Artwork

Sukanya Nag, Deepsikha Bhattacharjee, Archisman Bhaumik, and Suman Deb(✉)

National Institute of Technology Agartala, Agartala, India
sumandebcs@gmail.com

Abstract. An approach for enhancing the quality of vectorised images for robotic artwork, that act as input for CNC device, is addressed in this paper. Attempt to upgrade the vectorisation method has been made by comparing and combining the available edge detection techniques. Accordingly the scope of improving plotter functionality to achieve a human-like drawing from the plotter has been discussed. In the era of automation, the XY plotter is a powerful tool capable of producing artwork without external assistance. The plotter relies on vectorised images and thus the initial stage of experimentation includes creation of accurate vectors from ordinary images. The resultant vectors are processed by the CNC plotter as substantial artwork on a drawing surface. The conversion mechanism for producing drawing objects from bitmap image is time consuming. The paper proposes a deep learning based model which allows robust vectorisation of these images following a multi-stage approach.

Keywords: Vectorisation · Deep learning · Convolutional Neural Networks · Edge detection · Spatial robotic artwork

1 Introduction

Inspiration behind this work lies in vector formation which is a crucial part of artwork creation. The XY plotter [14] is the CNC device used to physically sketch the input image onto the drawing surface. A recurring problem of using ordinary bitmap images as input for the plotter is the blurring of the image. Bitmap images use pixels for representation which when enlarged can result in the distortion of the image. This causes a hindrance in acquiring clean output sketches as outputs from the device. As a result, vectorised images are chosen as inputs since here, the edges and boundaries are represented by individual vectors or Bezier curves. In case of a vectorised image, the image depends upon the equation of curves and not on quantity of pixels. This is why, even on scaling the images infinitely, we get a clear and distortion free image. A vectorised image constructed from a base image which could either be a hand-drawn sketch or photograph from a camera can be used as the input which is to be fed to the XY plotter.

S. Mekhilef et al. (Eds.): ICEEE 2022, LNEE 893, pp. 392–405, 2022.
https://doi.org/10.1007/978-981-19-1742-4_33

Since vector images are a common input source for XY plotters, it becomes significant to produce accurate and efficient vectors while working with similar CNC devices. The underlying logic of vectorisation is extracting the object boundaries from an image and converting them into vectors or mathematical curves having directions. Edges are the sudden changes of discontinuities in an image [17]. Any change in lighting conditions, shaded regions or texture can result in edge formation. A substantial segment of computer vision, image processing and in turn, vectorisation, is the concept of edge detection. In most cases, the initial stage of information retrieval from an image is edge detection. The process notably reduces the amount of data to be processed and filters out unnecessary information while preserving important structural features of the image. The classical edge detection mechanisms [7,9,16,20,23] are categorised into either Gradient or Gaussian. A number of edge detection techniques are available but each have their own drawback. The qualities of edges detected by the classical techniques are greatly dependent on external factors such as lighting conditions, noise, objects having similar intensities and edge density. Edge detection can be a complex and time consuming process especially for inputs involving presence of noise. Analysing the connected components of the image can become challenging using edge detection in situations involving noise. False edges may get detected which interferes with the outcome of the experiments. This is tremendously problematic while dealing with spatial artwork since every small detail adds up to a significant output. The variety of operators available do not produce the desired output after hardware implementation. The steps include filtering, enhancement and detection but the final result is a mediocre localisation of edges which does not meet the required criteria.

The approach presented in this paper targets these connected components and attempts to remove undesired noise in order to retain the necessary details. In this paper, we are interested in a comparison between Canny, Sobel and HED edge detection methods and present the suggested CNN based algorithm that properly synchronises with the X-Y plotter hardware set up. The mentioned algorithm attempts to incorporate the required features of the classical edge detectors in a Deep Learning model to favour the production of spatial robotic artwork. As the physical component of the set up, an X-Y plotter is used which is a CNC device that works on two stepper motors, an L293D driver and Arduino Uno microcontroller. The plotter uses a pen to draw across a plane surface. Vectorised images are fed to the plotter and the resultant image is drawn by the device itself. Later in this paper, a comparison is shown between MSED, which is the proposed algorithm, and the previously known techniques when used individually. The comparison should be sufficient to prove the efficiency of the proposed method of edge detection for spatial robotic artwork.

2 Literature Survey

Analysis of digital images can be simplified with the help of edge detection which deals with identifying pixels with high variations in intensity [17]. Edge detection is a primary step in measuring object size, isolating a desired object, object

recognition and classification. With the advancement of small devices and robust computation system, edge detection plays a vital role in image intelligence. If we take into account the human brain, it divides objects and its surroundings by colour. When we discuss image intelligence, we refer to the ability of the system to mimic the human brain in distinguishing objects in its surroundings [25]. An ideal system can accurately distinguish among all kinds of edges. There are several edge detection algorithms currently in use. The most popular among these edge detection algorithms, Canny [7] has certain drawbacks. Since Canny, most of the successive methods have been using non-maximum suppression [6] as the ultimate process in edge detection. A significant problem that was failed to be addressed in the classical methods is the ambiguity in edge detection leading to subjective contours [12]. The edges are always challenged by gradient colour, even proper identification of edges is ambiguous due to close colour representation. The parameters reducing amount of the least ambiguous edges of a detection technique could be considered as the significant ones. The concept of removal of the background, which is based on edge detection and contours, from the foreground has also not been touched upon in the classical edge detection techniques. It is proven to be of use in video calls, determining positions of obstacles in autonomous vehicles, and so on. To overcome these barriers in the path of efficient edge detection, various researchers have provided possible solutions aided by Convolutional Neural Network (CNN) and Deep Learning architectures, like HED [26], DeepEdge [4], DexiNed [21], etc. [10,15]. Although the CNN based schemes outperform the classical edge detectors, they possess a higher computational cost. Since edge detection is a rather simple job in comparison to image segmentation or object detection, the complicated architecture and enormous amount of parameters may be unnecessary. To reduce the complexity, few algorithms were proposed which generated thin edge-maps or good edges without previous training or fine tuning process [21] or by implementing traditional method inspired frameworks [24].

3 Edge Detection for Vector Creation

Edges are the locations in a digital image where there is a sharp change of intensity. A unit change from one pixel to the next can be considered as an ideal edge. Edges are classified into the following types:

- Horizontal edges
- Vertical edges
- Diagonal edges

  Edges are useful for segmentation, registration, and identification of objects [18]. In image analysis, a problem of fundamental importance is the detection of these edges. Edge detection [17] is defined as the technique of image processing based on regions of discontinuity. Classical edge detection methods are categorised into two types:
- Gradient

- Gaussian
  Gradient is used to determine the first-order derivatives of the intensity in the image whereas Gaussian methods are implemented to find the second-order derivatives. The process in general, is comprised of a series of fundamental steps that are as follows:
- Image smoothing: suppresses the noise without destroying true edges as shown in Fig. 1
- Detection of edge point: determining the pixels that should be retained while discarding the ones that contribute to noise
- Edge localisation: determines the exact location of the required edge
  Many factors are taken into consideration while performing edge detection, such as noise, brightness of the image, corners and edge-like features.

3.1 Sobel Edge Detection

The Sobel operator or filter [16], also known as Sobel-Feldman operator, works on the principle of discrete differentiation to compute the gradient approximation of image intensity function. Two 3×3 kernels are taken to be convolved with the input image which calculates the horizontal and vertical derivative approximations respectively.

Although the Sobel Edge Detector is beneficial for simple and time efficient computation and detection of smooth edges, it is highly sensitive to noise and fails to give appropriate results for thick and rough edges. Conclusively, it is not considered to be very accurate in edge detection.

3.2 Canny Edge Detection

Canny Edge Detector [7] is Gaussian-based and not susceptible to noise. Image features are extracted without affecting the feature. Canny edge detector has an advanced algorithm which is derived from the previous work on Laplacian of Gaussian operator.

Plus points held by the Canny operator are mainly good localization, extraction of features without altering them and minimised noise sensitivity; whereas false zero crossing, complex computation and higher time consumption attribute to the limitations possessed by this edge detector.

3.3 Holistically-Nested Edge Detection

Holistically-Nested Edge Detection, [26] or HED, is an end-to-end system that uses a trimmed VGG-like convolutional neural network for an image to image prediction task. HED automatically learns rich hierarchical representations that are important in order to determine the edge or object boundaries.

The edge map produced by this algorithm performs better at preserving object boundaries in comparison to the Canny Edge Detector.

Canny edge detection can be overtaken by Holistically Nested edge detection in applications where the environment and lighting conditions are unknown or uncontrollable, as the former requires preprocessing steps, manual tuning of parameters, and often does not perform well on images captured using varying lighting conditions. However, on the downside, HED is more expensive than Canny in terms of computational cost. For real-time performance, Canny would just require a CPU but HED would need a GPU for the same purpose.

4 Experiments on Edge Detectors

Here we have taken a standard image representing the desired category of spatial robotic artwork. To achieve expected results, some pre-processing steps could be carried out according to the necessity, before applying the Canny/Sobel/HED operator.

Image smoothing is one of the common pre-processing steps in various edge detection algorithms and could be achieved using filters such as Gaussian or Median blur. Traditional Canny algorithm makes use of Gaussian blur which seems to eliminate lot of information along with the noise. The edge details get smoothened significantly. However, application of Median blur does not create the same problem as edge details are retained even after removal of noise, as can be inferred from Fig. 1.

Following the preprocessing step, Canny operator is applied, followed by removal of noise and then vectorisation of the output. The process is repeated with Sobel operator and HED as shown in Fig. 3.

Fig. 1. Two of the techniques used for image smoothing

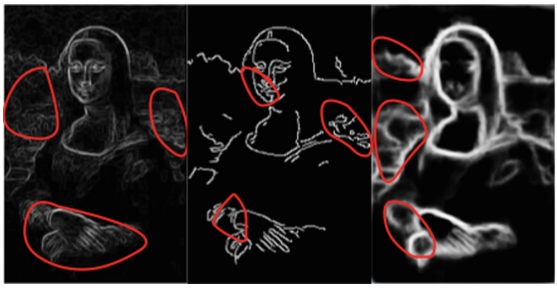

Fig. 2. Noise and unnecessary details retained in (a) Sobel (b) Canny and (c) HED detected images make them unsuitable for our purpose

Comparisons on Edge Detectors. As it can be inferred from the above experiments in Fig. 2, the edges formed from Sobel edge detection have low intensity and thus some significant edges get removed while vectorisation. The output from HED contains edges having better intensity than the Sobel output but they are not as sharp as the Canny output. Canny edge detection algorithm produces sharper edges in the vectorised output in comparison to Sobel and HED. These vectorised outputs will not produce an efficient artwork as unwanted objects are still retained. The formation of double lines will also add up to production of unwanted edges while drawing with the XY plotter. The presence of noise, as shown encircled in the images, is a hindrance in achieving the desired effect of human-like drawing. The vector images produced from these algorithms do not meet the desired need and hence, we aim to implement an algorithm that would give more robust results.

5 Multi-Stage Edge Detection

The Multi-Stage Edge Detection (MSED) algorithm introduced in this paper attempts to combine the refined aspects of classical edge detection algorithms with a CNN model assuring the production of robust vector images. The approach followed in MSED can be broken down into five steps :

1. Image pre-processing
2. Application of classical algorithm (Sobel, Canny or a combination of both)
3. Noise Removal
4. Application of a CNN model
5. Vectorisation

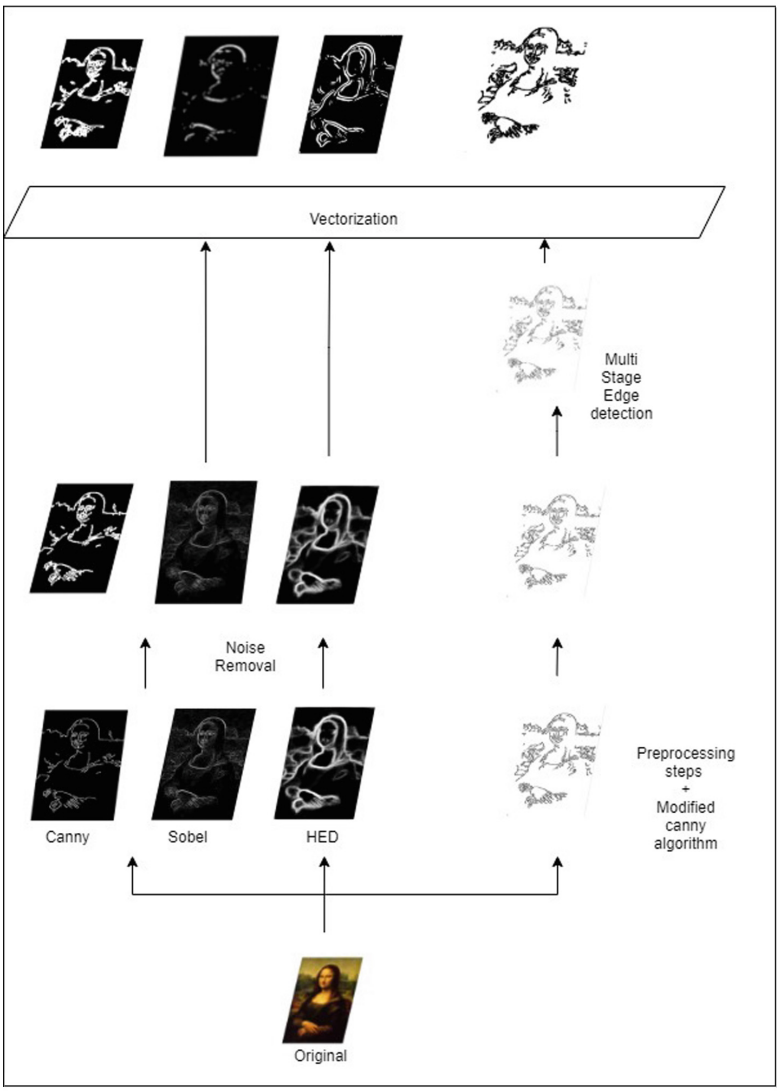

Fig. 3. The algorithm flowchart for MSED and related comparison

5.1 Image Preprocessing

Image pre-processing is an optional step which depends on the quality of the image. This step mostly includes the primary edits which refine the input image before application of edge detection techniques. Contrast and brightness of the image are the targeted settings which, when adjusted properly, enhance the quality of the image before the further techniques can be applied to it.

5.2 Application of Classical Algorithm

Any of the two mentioned classical techniques can be used as the first step of MSED. However, since Canny has proven to be more efficient in producing sharp edges than Sobel, the former is preferred. The Frei-Chen gradient calculation algorithm [2] replaces the common gradient calculation in Canny edge detection technique, which makes up for the deficiency of the latter. The calculation method is shown below:

$$\frac{1}{2+\sqrt{2}} \begin{bmatrix} -1 & 0 & 1 \\ -\sqrt{2} & 0 & \sqrt{2} \\ -1 & 0 & 1 \end{bmatrix} \; rowgradient \tag{1}$$

$$\frac{1}{2+\sqrt{2}} \begin{bmatrix} 1 & \sqrt{2} & 1 \\ 0 & 0 & 0 \\ -1 & -\sqrt{2} & -1 \end{bmatrix} \; columngradient \tag{2}$$

5.3 Noise Removal

Filtering out unnecessary image data is a standard process used for almost every image processing purpose. Filters are used for achieving this by removing noise from images while preserving the details of the same. Image denoising [5] is the process of noise removal from an image.

Noise originates by several ways such as image acquisition, transmission, and compression. Since presence of noise in an image leads to loss of valuable information, it is essential to remove the noise and recover the original image from the degraded images where getting the original image is important for robust performance or in cases where filling the missing information is very useful.

Several filters are available for noise removal, used according to the type of noise and the requirement. Broadly, the filters are classified under two domains - Time and Frequency. Mean, Gaussian and Median, belonging to the Time Domain, are few of the commonly used filters in pre or post processing of images for removal of noise.

The original Canny algorithm uses the Gaussian filter, making the edge detail seem blurry whereas our expected results require intense edges. This can also cause creation of false edges during vectorisation. To solve this problem, we pick the Median filter for our algorithm for preserving the required details of the image while removing the noise as evident from Fig. 1. In the median filter, which can be classified as a low-pass filter, a window slides along the image, and the median intensity value of the pixels within the window becomes the output intensity of the pixel being processed outputs the median intensity value of the pixels. This step provides a better result in comparison to the Gaussian counterpart.

5.4 Application of a CNN Model

A human brain is intelligent and thus can easily process any form of data that is provided. On seeing an image with multiple components, the human can identify the various objects and their connected parts. When a robot tries to mimic this natural human behaviour, the concepts of image segmentation, object recognition and analysis of connected components are crucial. These concepts are taken into consideration while building the CNN model.

The benefit of using a Convolutional Neural Network (CNN) based Deep Learning model [1] in edge detection is that it generates intense edges without the need of any post processing steps. CNN is applied in the later stage of the MSED algorithm to generate the desired image. Even after the detection of edges using multiple edge detectors, most of the times various post processing steps have to be applied in order to generate the finer edges and reduce the quantity of noise, before it could be used in concerned applications. However, incorporation of CNN based models has helped in obtaining the desired results. Usage of convolutional network layers provides two major advantages:

- Parameter sharing [22]
- Sparsity of connections [8] Parameter sharing is motivated by the observation that a feature detector, like a horizontal edge detector, that's useful in one part of the image could be probably useful in another section of the image as well. This implies that, if one figures out, say a 3×3 filter for detecting horizontal edges, the same can be applied at another region, for example, the lower right-hand corner, and then another position over to the left, and so on. Hence, the feature detectors can use the same parameters in lots of different positions in the image. Application of CNN has proven to be very efficient as it can compute all features automatically, hence reducing the burden of manual feature extraction. Moreover, feature detection using CNN results in more accurate and precise outcomes when compared to manual feature compilation methods. For the proposal in this paper, the connected component analysis takes an important role. The underlying concept is to identify the pixels that belong to a particular object by exploiting the pixel neighbourhood. The input is a binary image in the form of a matrix where pixels representing background can be represented by false(0) values and the components in the foreground have true(1) values. Connected objects in the foreground are identified using two rules followed throughout the process. Firstly, jumps (traversing through pixels to find connected objects) are allowed only along a row or a column. The diagonal jumps are not allowed. In the second rule it is stated that in a sequence of multiple jumps, a jump in row and column direction can be made only once and it has to be orthogonal. The importance of analysis of connected components is realised while vectorising the image. An input consisting of a clean outline of the components in the image is a necessity for a refined output of the Multi-Stage edge detection algorithm.

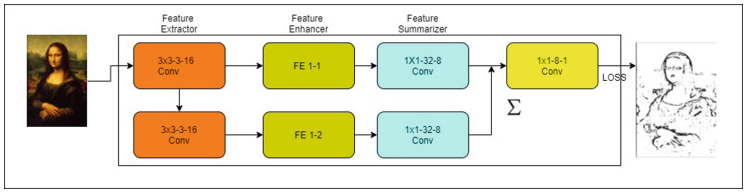

Fig. 4. Architecture of the CNN model

Our CNN-based model consists of Feature Extractor, Feature Enhancer and Feature Summarizer, as shown in Fig. 4, which roughly corresponds to gradient, low pass filter and pixel connection respectively in the classical edge detection techniques. The feature extractor is a 3×3 convolutional neural network layer. 16 feature channels are used which are initialized with the 16 directional gradient kernels. Rest of CNN layers are initialized with zero-mean, 0.01 standard deviation Gaussian and zero biases. In the Feature Enhancer module, dilated convolutions are used for creating multiple scale filtering. A larger receptive field can be captured with fewer parameters. This module is named Enhancer as it extracts object information along with the edges. The Feature Summarizer module, which is the last one, summarizes the features generated by the Enhancer modules and produces the final edges. Eight 1×1 convolutional layers are applied along with a sigmoid activation function.

5.5 Vectorisation

The final step of the entire technique leading up to the finished output is the creation of desired vector images [3]. Since the motivation behind the work stated in the paper was derived from the creation of spatial robotic artwork primarily for the use of computer controlled devices [14], this step serves as evidence to the entire work that has been done so far.

In computer graphics, vectorisation, also called image tracing or raster-vector conversion is the conversion technique of raster graphics into vector graphics. Bitmap images are represented through pixels because of which it tends to resize improperly and get pixelated on being enlarged. This problem is solved using vector graphics where edges in the image are represented through equations of Bezier curves or curved paths instead of pixels. Bezier curves display a prominent role in computer graphics, especially in raster to vector conversion as the dynamic nature of these curves make them appropriate for defining boundaries. They also help in refitting deformed shapes in raster images which often become distorted in quality when magnified to a greater extent.

During experimentation for this paper, Inkscape has been used for vectorisation. G-code tools are provided with Inkscape to allow G-code generation from vector images. G-code is the most popular programming language which is used for controlling machines such as a 3D-printer or, in this case, an XY plotter. It has many variants yet a large number of them follow some common set of rules.

The use of the Arduino for the XY plotter is facilitated through the built-in GRBL [19] firmware. GRBL then uses the G-code for controlling the hardware.

Single line g-code blocks are sent to GRBL, which is an open-source software designed for three-axis control of stepper-motor based CNC machines, awaiting for an acknowledgement. GRBL sends an acknowledgement once processing of the block is completed and there is room in the buffer.

GRBL receives ASCII commands and G-code and performs smooth linear and circular interpolation for the stepper motors. It operates through the Arduino serial port interface and requires a constant stream of g-code commands sent through a computer. The single g-code blocks are accepted and processed, followed by a carriage return. It returns an 'ok' or 'error:X' message after the block is processed and is ready for further information. The entire process of getting the desired output on the drawing board after feeding the bitmap image is shown in the Fig. 5 below.

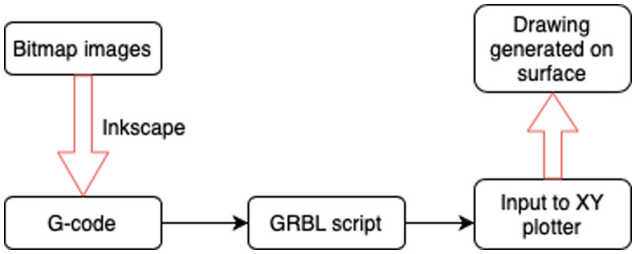

Fig. 5. Input bitmap image to final drawing

6 Experiments on MSED

The implementation of our model is achieved through Pytorch. The Gaussian distribution algorithm is used to initialize all the filter weights except in the first stage of edge detection, which is Feature Extractor 1. The mean is initialized to 0 and standard deviation to 0.01, and the biases are initialized with 0. The hyper-parameters are: mini-batch size = 1, learning rate = $(1e{-}2)$, weight decay = $(5e{-}4)$, momentum = 0.9, and training epochs = 120. For every 10 epochs, the learning rate is divided by 10. We utilize the SGD solver for optimization. All the experiments are conducted with the assistance of NVIDIA GeForce 2080Ti GPU with 11 G memory. The steps mentioned in Sect. 5 are followed sequentially till step D to obtain the left image, shown in Fig. 6. Post vectorisation as on Step E, the image on the right is produced which is the ultimate output of the new algorithm. The experimental output has shown significant improvement from the existing algorithms while dealing with reproduction of spatial artwork using XY robotic plotters.

Fig. 6. The final output after MSED and vectorisation respectively

Finally the image, which has been edge detected followed by vectorisation, on being uploaded to the robotic plotter, produces an artwork on the drawing surface with the assistance of a pen. Considering the fact that the XY plotter is a project still awaiting completion in terms of hardware assembling, we provide sample images, Fig. 7, that depict the expected outcome on the surface.

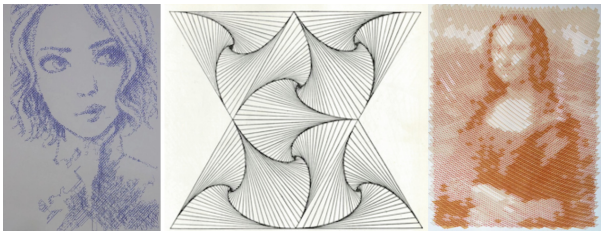

Fig. 7. Sample vectorised artwork drawn on surface Source: Adapted from [11,13]

7 Conclusion

The proposed Multi-Stage Edge Detection technique, based on a blend of classical and convolutional neural networks, has given expected outputs which are computationally faster and qualitatively better for vectorisation, in comparison to the existing algorithms.

While emphasising on spatial robotic artwork, the images formed by the application of the method we formulated are comparable to an actual human-drawn sketch which usually consists of outlines of the objects and is devoid of any noise and unwanted objects. Our technique helped in obtaining the desired

form of sketch we strived to achieve through the plotter device. This outcome will provide the XY robotic plotter with an additional functionality of reproducing the artwork, fed to it as input, in form of a sketch.

Hence this approach demonstrates promising results and has the potential to further contribute greatly in overcoming the targeted working limitations of computer numerical controlled devices.

References

1. Ahmed, A., Byun, Y.-C.: Edge detection using CNN for roof images, pp. 75-78 (2019)
2. Apdilah, D., Simargolang, M., Rahim, R.: A study of Frei-Chen approach for edge detection. Int. J. Sci. Res. Sci. Eng. Technol. **3**, 59–62 (2017)
3. Bera, A.: Fast vectorization and upscaling images with natural objects using canny edge detection. In: vol. 3, p. 4 (2011)
4. Bertasius, G., Shi, J., Torresani, L.: DeepEdge: a multi-scale bifurcated deep network for top-down contour detection (2015)
5. Buades, A., Coll, B., Morel, J.M.: A review of image denoising algorithms, with a new one. Multiscale Model. Simul. **4**(2), 490–530 (2005)
6. Canny, J.: Finding edges and lines in images. Theory of Computing Systems Mathematical Systems Theory, p. 16 (1983)
7. Canny, J.: A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. **6**, 679–698 (1986)
8. Changpinyo, S., Sandler, M., Zhmoginov, A.: The power of sparsity in convolutional neural networks (2017)
9. Chaple, G.N., Daruwala, R.D., Gofane, M.S.: Comparisons of Robert, Prewitt, Sobel operator based edge detection methods for real time uses on FPGA. In: 2015 International Conference on Technologies for Sustainable Development (ICTSD), pp. 1–4 (2015)
10. Chaudhary, D.K., Lal, R., Kashyap, N., Choudhury, T.: Hybrid edge detection technique for digital images, pp. 1116–1121 (2016)
11. Das, A.K.: How to make an Arduino drawing machine? November 2018
12. Davis, S., Jernigan, E.: Ambiguous edge detection leads to subjective contours. Percep. Psychophys. **31**, 93–94 (1982)
13. Good, I.J., Vite, M.: Science in the flesh in cybernetics, arts and ideas by Reichardt, J. 1971 (1968)
14. Jegan, R.R., Gnanasundaram, E., Gowtham, M., Sivanesan, R., Thiyagarajan, D.: Modern design and implementation of XY plotter, pp. 1651–1654 (2018)
15. Yang, M., Shan, Y., Huang, T., He, J., Zhang, S.: Bi-directional cascade network for perceptual edge detection
16. Kanopoulos, N.: Design of an image edge detection filter using the Sobel operator. IEEE J. Solid State Circuits **23**(2), 358–367 (1988)
17. Marr, D.C., Hildreth, E.: Theory of edge detection. Proc. Roy. Soc. London **207**, 187–217 (1980)
18. Rosenfeld, A., Thurston, M.: Edge and curve detection for visual scene analysis. IEEE Trans. Comput. **C-20**(5), 562–569 (1971)
19. Sarguroh, S.S., Rane, A.B.: Using GRBL-Arduino-based controller to run a two-axis computerized numerical control machine, pp. 1–6 (2018)

20. Seif, A., Salut, M.M., Marsono, M.N.: A hardware architecture of Prewitt edge detection. In: 2010 IEEE Conference on Sustainable Utilization and Development in Engineering and Technology, pp. 99–101 (2010)
21. Soria, X., Riba, E., Sappa, A.: Dense extreme inception network: towards a robust CNN model for edge detection (2020)
22. Terry, J.K., Grammel, N., Hari, A., Santos, L., Black, B.: Revisiting parameter sharing in multi-agent deep reinforcement learning (2021)
23. Wang, X.: Laplacian operator-based edge detectors. IEEE Trans. Pattern. Anal. Mach. Intell. **29**, 886–890 (2007)
24. Wibisono, J.K., Hang, H.-M.: Traditional method inspired deep neural network for edge detection. In: 2020 IEEE International Conference on Image Processing (ICIP) (2020)
25. Wu, Q.X., McGinnity, M., Maguire, L., Belatreche, A., Glackin, B.: Edge detection based on spiking neural network model, pp. 26–34 (2007)
26. Xie, S., Tu, Z.: Holistically-nested edge detection. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (2015)