ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

PhD. THESIS

İsmail Can YILMAZ

ULTRAFAST PATTERN BASED LINE DETECTOR FOR REAL TIME IMAGE PROCESSING

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

ADANA-2022

In Loving Memory of My Mom Kamile

ABSTRACT

PhD. THESIS

ULTRAFAST PATTERN BASED LINE DETECTOR FOR REAL TIME IMAGE PROCESSING

İsmail Can YILMAZ

ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES
DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

Supervisor : Prof. Dr. Mustafa GÖK
Co-supervisor : Asst Prof. Dr. İbrahim Cem BAYKAL
Year: 2022, Pages: 76
Jury : Prof. Dr. Mustafa GÖK
: Prof. Dr. Ulus ÇEVİK
: Prof. Dr. Mutlu AVCI
: Asst. Prof. Dr. Hüseyin AFŞER
: Asst. Prof. Dr. Zehan KESİLMİŞ

Line detector algorithms are used for feature extraction, segmentation, pattern recognition, matching and object detection in computer vision applications such as detection of road lanes and power lines. The fastest line detector so far published in the literature is presented in this thesis. This line detector algorithm is 14.88 times faster than the next fastest line detector, the EDLines, and 94.7 times faster than the line segment detector and more reliable than the Standard Hough Transform. An ingeniously simple method for detecting and combining patterns with advantage of several look-up tables is introduced to recognize and fit straight-line patterns. As the first step, it recognizes any possible 4 × 4 pixel line patterns among the binary edge pixels and then uses the slope and the connection points look-up tables to decide whether the connected patterns form a line or not. This algorithm is implemented and tested in MATLAB to compare the speed performance. MATLAB executable file is created by compiling C++ code. The experimental results are presented as graphics and tables in terms of the speed. According to the results, the proposed algorithm is an ultrafast line detector for real time image processing. This algorithm is especially designed for high definition images.

Key Words: Line Detector, Feature Extraction, Segmentation, Pattern Recognition, Image Processing, Matlab

ÖZ

DOKTORA TEZİ

GERÇEK ZAMANLI GÖRÜNTÜ İŞLEME İÇİN ULTRA HIZLI DESEN TABANLI ÇİZGİ DEDEKTÖRÜ

İsmail Can YILMAZ

ÇUKUROVA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK – ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

Danışman : Prof. Dr. Mustafa GÖK
İkinci Danışman : Dr. Öğr. Üyesi İbrahim Cem BAYKAL
Yıl: 2022, Sayfa: 72
Jüri : Prof. Dr. Mustafa GÖK
: Prof. Dr. Ulus ÇEVİK
: Prof. Dr. Mutlu AVCI
: Dr. Öğr. Üyesi Hüseyin AFŞER
: Dr. Öğr. Üyesi Zehan KESİLMİŞ

Çizgi algılama algoritmaları, yol şeritlerinin ve enerji hatlarının tespiti gibi bilgisayarlı görü uygulamalarında özellik çıkarma, segmentasyon, örüntü tanıma, eşleştirme ve nesne tespiti için kullanılır. Düz çizgi modellerin tanınması ve oluşturulması için çeşitli arama tablolarının avantajıyla çizgi desenlerini algılayan ve birleştiren ustaca basit bir yöntem tanıtıldı. Bu yöntem, ilk adım olarak, ikili kenar pikselleri arasında olası 4 × 4 piksel çizgi desenlerini tanır ve ardından komşu desenlerin bir çizgi oluşturup oluşturmadığına karar vermek için eğim ve bağlantı noktaları arama tablolarını kullanır. Bu algoritma, hız performansını karşılaştırmak için MATLAB'da uygulanmış ve test edilmiştir. MATLAB çalıştırılabilir dosyası, C++ kodunun derlenmesiyle oluşturulur. Deneysel sonuçlar hız açısından grafik ve tablolar halinde sunulmuştur. Elde edilen sonuçlara göre bu önerilen algoritma, gerçek zamanlı görüntü işleme için ultra hızlı bir çizgi dedektörüdür. Bu algoritma, özellikle yüksek çözünürlüklü görüntüler için tasarlanmıştır.

Anahtar Kelimeler: Çizgi Dedektörü, Özellik Çıkarma, Segmentasyon, Örüntü Tanıma, Görüntü İşleme, Matlab

EXTENDED SUMMARY

Although line detection in image processing is an important research topic, line detection is still an unsolved problem in computer vision. Every year, tens of studies about line detection are averagely published in the literature.

When line detection is concerned, the famous Hough transform is the first algorithm that comes to mind. The HT algorithm converts every edge pixel into sinusoids and superimposes these sinusoids to find the intersection points. In Hough space, these intersection points create the peak points give us the equations for the lines. However, this algorithm does not actually work perfectly in an image. Because, there are always 8 pixels surrounding a single pixel in the discrete space, and these neighbour 8 pixels create flat peaks instead of a spike.

The other famous line detection algorithms are the LSD and the EDlines. The fact that, the EDLines algorithm is based on the improved LSD by using an its own faster edge detector. As a result, they both have same result performance but the LSD is slower than the EDLines.

The LSD algorithm scales the input image to reduce noise as a first step. And then, the gradient magnitude of each edge pixel is calculated with a 2×2 kernel matrix. Thirdly, the edge pixels with weak gradient are eliminated by using thresholding method. Finally, the adjacent pixels with similar gradient are grouped by using the region (rectangular) technique with a contario approach for line segment detection.

The EDLines algorithm has the smoothing step to reduce noise in the image and the gradient calculation step as the LSD algorithm. After gradient calculation, the connected peak pixels, anchors, are linked as a pixel chain to draw edge pixels of the input image. Finally, the EDlines algorithm uses the least squares fitting to fit lines segments onto each pixel chain and these line segments are validated by using the number of false alarms. There are two major shortcomings of the EDlines

algorithm; detecting false edges in a noisy images and dividing arcs into too many small lines.

In this thesis, a novel, completely original and ultrafast line detector based on recognition of the 4 × 4 pixel line patterns is presented. This proposed algorithm is the fastest line detector so far published in the literature. Briefly, the Canny methods or the EDLines edge detector is applied to an image for the binary edge image and the edge image is divided to 4 × 4 pixel windows. These 4 × 4 pixels are recognized with the help of a 64 KB look-up table. In this look-up table, the binary, slope, start and end point information of the 4 × 4 pixel line patterns are available. There are defined 110 4 × 4 pixel line patterns with 8 different slope and 12 tip points value. After the recognition of the line patterns for the segmentation, they must meet conditions in the slope and the connection points look-up tables. Finally, the result of segmentation gives the start and end point of the lines.

As a consequence, the fastest line detector based on pattern recognition is presented according to the graphically and tabulated test results in terms of speed performance.

GENİŞLETİLMİŞ ÖZET

Görüntü işlemede çizgi tespiti önemli bir araştırma konusu olmasına rağmen, bilgisayarlı görüde çizgi tespiti hala tam olarak çözülmemiş bir problemdir. Literatürde, her yıl ortalama olarak çizgi tespiti ile ilgili onlarca çalışma yayınlanmaktadır.

Çizgi tespiti denilince akla ilk gelen algoritma ünlü Hough dönüşümüdür. HT algoritması, her kenar pikselini sinüzoitlere dönüştürür ve kesişme noktalarını bulmak için bu sinüzoitleri üst üste bindirir. Ancak bu algoritma aslında bir görüntüde tam olarak çalışmaz. Hough uzayında, bu kesişme noktaları tepe noktalarını oluşturur ve bize doğruların denklemlerini verir. Ancak bu algoritma aslında bir görüntüde tam olarak çalışmaz. Çünkü ayrık uzayda tek bir pikseli çevreleyen her zaman 8 piksel vardır ve bu komşu 8 piksel, bir sivri uç yerine düz tepeler oluşturur.

LSD ve EDlines, diğer ünlü çizgi algılama algoritmalarıdır. EDLines algoritmasının, kendine ait daha hızlı bir kenar detektörü kullanarak geliştirilmiş LSD'ye dayandığı gerçektir. Sonuç olarak, her ikisi de aynı sonuç performansına sahiptir ancak LSD, EDLines'den daha yavaştır.

LSD algoritması, ilk adım olarak gürültüyü azaltmak için giriş görüntüsünü ölçeklendirir. Daha sonra, her bir kenar pikselinin gradyan büyüklüğü 2 × 2 çekirdek matrisi ile hesaplanır. Üçüncü adım olarak, eşikleme yöntemi kullanılarak zayıf gradyanlı kenar pikseller elimine edilir. Son olarak, benzer gradyanlı bitişik pikseller, çizgi segmenti tespiti için bir kontario yaklaşımıyla bölge (dikdörtgen) tekniği kullanılarak gruplandırılır.

Bu tezde, 4 × 4 piksel çizgi desenlerinin tanınmasına dayalı yeni, tamamen orijinal ve ultra hızlı bir çizgi detektörü sunulmaktadır. Önerilen bu algoritma, şimdiye kadar literatürde yayınlanan en hızlı çizgi detektörüdür. Kısaca, ikili kenar görüntüsü için bir görüntüye Canny yöntemleri veya EDLines kenar detektörü

uygulanır ve bu kenar görüntüsü 4 × 4 piksel pencerelere bölünür. Bu 4 × 4 pikseller, 64 KB'lık bir arama tablosu yardımıyla tanınır. Bu arama tablosunda, 4×4 piksel çizgi desenlerinin ikili değeri, eğim, başlangıç ve bitiş noktası bilgileri mevcuttur. 8 farklı eğim ve 12 uç nokta değerine sahip 110 tane farklı 4 × 4 piksel çizgi desenleri tanımlanmıştır. Çizgi desenlerinin tanınmasından sonra, segmentasyon için eğim ve bağlantı noktaları arama tablolarındaki koşullarının sağlanması gerekmektedir. Son olarak, segmentasyonun sonucu, çizgilerin başlangıç ve bitiş noktalarını verir.

Sonuç olarak, hız performansı açısından grafik ve tablolu test sonuçlarına göre örüntü tanımaya dayalı en hızlı çizgi detektörü sunulmaktadır.

ACKNOWLEDGEMENTS

I would like to present my endless gratitude to my supervisor Prof. Dr. Mustafa GÖK and my co-supervisor Asst Prof. Dr. İbrahim Cem BAYKAL for their unlimited support, mentoring, advice, criticism, guidance, patience, motivation and understanding for the successful completion of this research work.

I wish to express my special thanks to Prof. Dr. Ulus ÇEVİK, Prof. Dr. Mutlu AVCI and Asst. Prof. Dr. Zehan KESİLMİŞ for their valuable suggestions and supports to accomplish my thesis.

I also wish to express my special thanks to Asst. Prof. Dr. Hüseyin AFŞER for accepting to be the member of examining committee for my thesis and for his valuable suggestions.

I would also thankful to my wife Senem YILMAZ and my sons Ali Yekta, Asım Emir and Aytuğ Kerem for their strong patience during the preparation of this study.

Finally, this study has been financially supported by Turkish National Science and Technology Foundation (TUBITAK) (Award No. 115C008).

CONTENTS **PAGE**

LIST OF TABLES **PAGE**

LIST OF FIGURES **PAGE**

LIST OF ABBREVIATIONS

AVX : Advanced Vector Extensions
ED : Edge Drawing
GUI : Graphical User Interface
HT : Hough Transform
LIDAR : Light Detection and Ranging
LSD : Line Segment Detector
LSF : Least Squares Fitting
MEX : MATLAB Executable File
NFA : Number of False Alarms
PHT : Probabilistic Hough Transform
SLAM : Simultaneous Localization and Mapping
STL : Standard Template Library

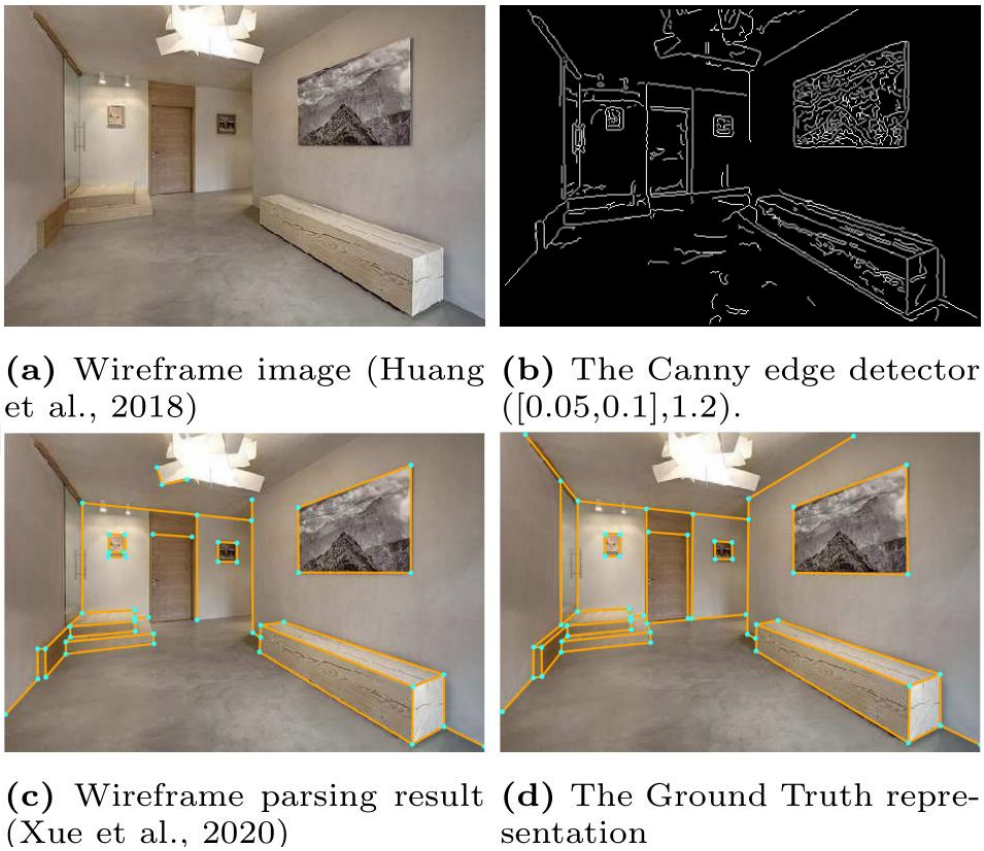
1. INTRODUCTION

The line detectors are introduced in the present chapter. Line detectors (Zhao et al., 2022; Deng et al., 2021; Vakhitov and Lempitsky, 2019) are very important topics in computer vison and recently used for lane detection (Andrade et al., 2019; Yeniaydin, 2019) as advanced driver assistance systems for autonomous vehicle, wireframe parsing (Xue et al., 2020; Zhou et al, 2019), shape descriptor (Lin et al., 2020), ellipse detection (Lu et al., 2019; Meng et al., 2020), noncontact cable force estimation with unmanned aerial vehicle (Tian et al., 2021) man-made environments (Zou et al., 2019; Zhang et al., 2019), recognition of dashed curves (Liu et al., 2022), light detection and ranging (Lidar) visual odometry (Huang et al., 2020), simultaneous localization and mapping (SLAM) system (Fu et al., 2019), vehicle detection (Tao et al., 2019), feature descriptor (Jiang et al., 2021), vanishing point detector (Wu et al., 2021), polygonal partition (Zhang et al., 2021; Li et al., 2020) and perspective distortion correction (Li et al., 2019).

Figure 1.1 shows the wireframe parsing example (Xue et al., 2020) on the wireframe dataset (Huang et al., 2018). Xue et al. briefly defines wireframe parsing as jointly detecting meaningful and salient line segments and junctions and uses three steps (i) line segment and junction generation, (ii) line segment and junction matching and (iii) line segment and junction verification in their wireframe parsing algorithm. According to them, line segment detection and junction detection remain challenging problems in computer vision.

In this chapter, popular line detectors in the literature such as Hough Transform (HT) and the variants of HT, Etemadi, Kovesi, line segment detector (LSD) and EDLines are defined. Hough Transform (HT) is initially described and then the variants of HT are presented.

The main objectives, contributions, and the organization of the thesis are provided at the end of the chapter.



(a) Wireframe image (Huang et al., 2018)

(b) The Canny edge detector ([0.05,0.1],1.2).

(c) Wireframe parsing result (Xue et al., 2020)

(d) The Ground Truth representation

Figure 1.1 The wireframe parsing example (Xue et al., 2020) on the wireframe dataset (Huang et al., 2018).

1.1. Background and Motivation

Line detectors (Yilmaz and Baykal, 2022) are basically divided into two types. First type requires edge detected images as their input and the other type have their own gradient/edge detectors. Most of the HT variants, Etemadi (Etemadi, 1982) and this proposed algorithm are in the first category and Burns (Burns et al., 1986), the line segment detector (LSD) (Von Gioi et al., 2008) and EDLines (Akinlar and Topal, 2013) are among the second.

Since the invention of the HT algorithm in 1962 (Hough, 1962), hundreds of survey (Illingworth and Kittler, 1988; Mukhopadhyay and Chaudhuri, 2015) have been published about different variants of the HT. Briefly, HT converts every edge

pixel into sinusoids and these sinusoids are superimposed to find the locations of peak points for the line equations. Unfortunately, the mathematical proof given in the continuous space does not work as efficiently in the discrete space because every single pixel is surrounded by an 8 neighbouring pixels. These neighbouring pixels create flat peaks instead of a spike (Van Veen and Groen, 1981) and these flat peaks (Brown, 1983; The MathWorks, 2022) create false lines so that the HT becomes unreliable (Baykal, 2019).

Relatively new HT articles (Guerreiro and Aguiar, 2012; Matas et al., 2000) improve the HT algorithms but these methods are slow and the results of the input tree image, given in Figure 1.2 are shown in Figure 1.3 and Figure 1.4.

Figure 1.2 The same “tree” image was used in the LSD’s article.

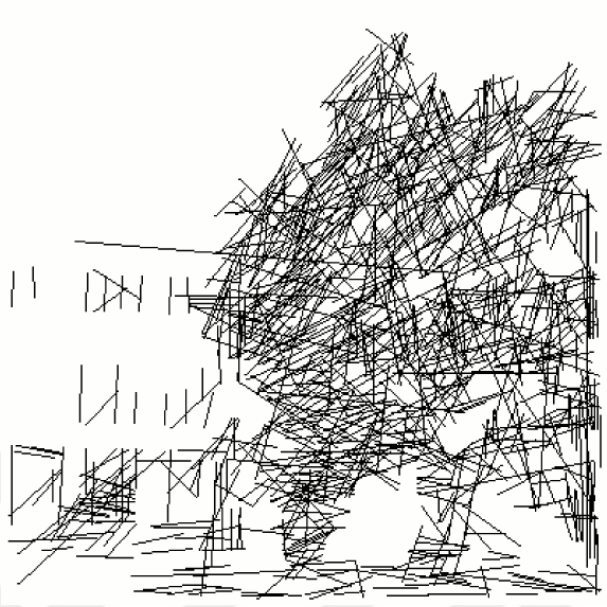


Figure 1.3 The result of the Probabilistic HT (PHT) (PI/180,15,20,10) 12.8 ms.

Figure 1.4 The result of the connectivity enforcing HT 17900 ms.

Recently LSD (Von Gioi et al., 2008) has become rapidly popular because of filling a gap for much-needed reliable and fast line detectors. The LSD article uses tree image in Figure 1.2, for performance comparison and the result is shown in Figure 1.5. There are many different line segment detector methods in the literature (Liu et al., 2020; Cho et al., 2017; Huang et al., 2018; Xue et al., 2019).

Figure 1.5 The result of LSD 28.9 ms.

The EDLines algorithm (Akinlar and Topal, 2013; Topal and Akinlar, 2012) is another popular line detector based on improving the LSD by using a faster edge detector. This algorithm firstly applies a low-pass filter to the image and then uses a Sobel operator to calculate the edge gradient. Largest points on this gradient are called anchors and these anchor points are used for chaining pixels to fit a line by the least-squares algorithm. In this algorithm, the Helmholtz principle is an optional method for line validation but false edges are detected in noisy images (Lu et al., 2015) because of the quality of the edge detector. To improve the EDLines algorithm, a Canny edge detector-based algorithm was proposed (Lu et al., 2015) but this algorithm is slower than the EDLines. Figure 1.6 shows the result of EDLines.

Figure 1.6 The result of EDLines 7.2 ms.

The simplest method of detecting lines is to fit a line to a group of connected edge pixels. One of the earliest examples of this approach is Etamadi's algorithm (Etemadi, 1982). In this algorithm, the pixels are judged to be symmetric or asymmetric and a simple line fitter is used among the symmetry pixels.

Despite being a simple algorithm, the code published by Etemadi runs unfairly slow because of poor implementation so that some improvements are applied to the code of this algorithm (GitHub, 2022). This new Etemadi implementation also has a graphical user interface (GUI) and is actually two times faster than LSD. If that code is written again by using standard template library (STL), it would be four times faster than LSD. Figure 1.7 shows the result of modified Etemadi.

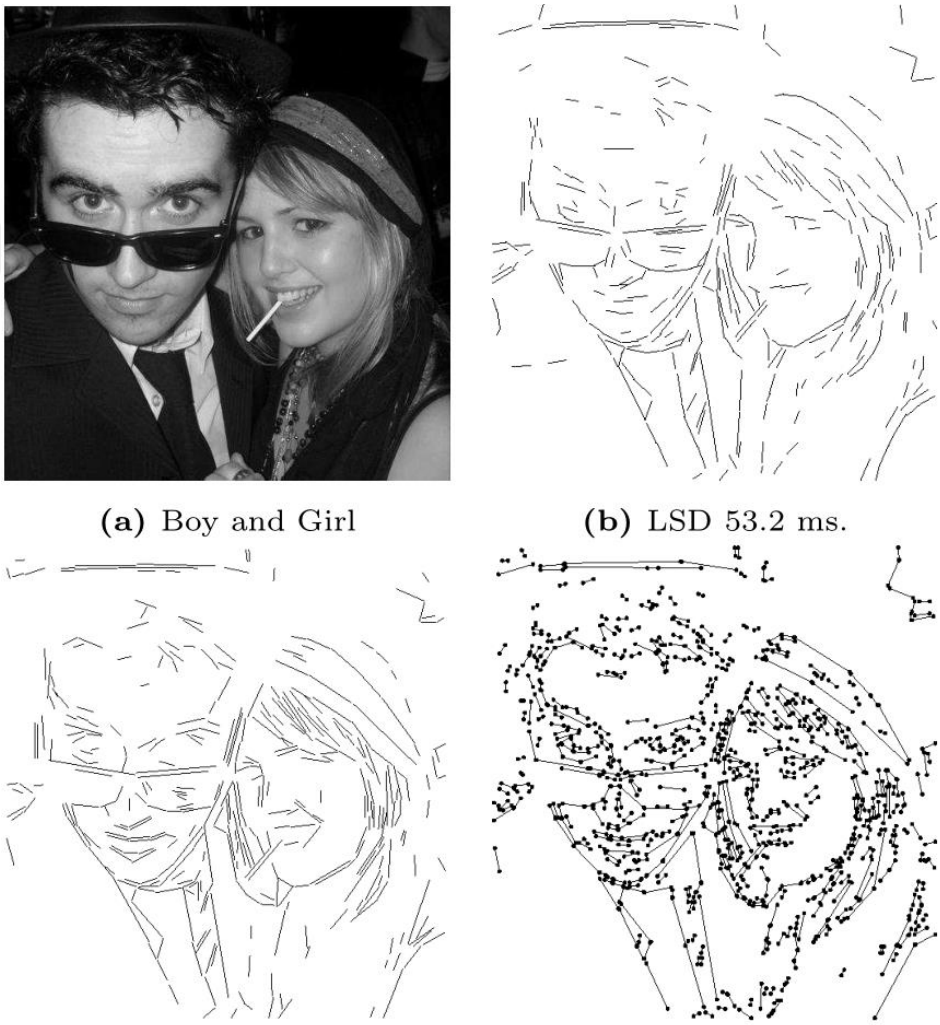
Figure 1.7 The result of Modified Etemadi's result 16.5 ms.

Essentially, the proposed algorithm is also a type of pixel chaining algorithm. The main difference is that in the proposed algorithm, an ingeniously simple method to chain 4 × 4 line segments and look-up tables to fit those segments to a line are used. Therefore, the proposed algorithm is far faster than the other popular algorithms. The proposed algorithm uses the Canny edge detector with parameters, ([0.05,0.1],1.2), in Figure 1.8. Figure 1.9 shows the result of the proposed algorithm which uses the edged image in Figure 1.8 as an input.

Figure 1.8 The result of the Canny edge detector ([0.05,0.1],1.2).

Figure 1.9 The result of the proposed algorithm 0.36 ms.

Figure 1.10 shows the comparison of LSD, EDLines and the proposed algorithm. According to the Figure 1.10 (b), (c) and (d), LSD and EDLines have the similar output and the proposed algorithm gives more details than LSD and EDLines.



(a) Boy and Girl **(b)** LSD 53.2 ms.

(c) EDLines 7.54 ms. **(d)** The Proposed A. 0.42 ms.

Figure 1.10 (a) The "boy and girl" image. (b) The result of LSD. (c) The result of EDLines. (d) The result of the proposed algorithm.

1.2. Literature Review

In the literature, the simple edge pixel chaining line detector code is Peter Kovesi's Matlab implementation (Kovesi, 2022). His implementation applies least-squares line fitting method among the chained pixel. In this implementation, the error threshold is used for fitting the line and the curves fitted to strictly lines are decided by user. Figure 1.11 shows the example of fitting lines to arc with different error thresholds.

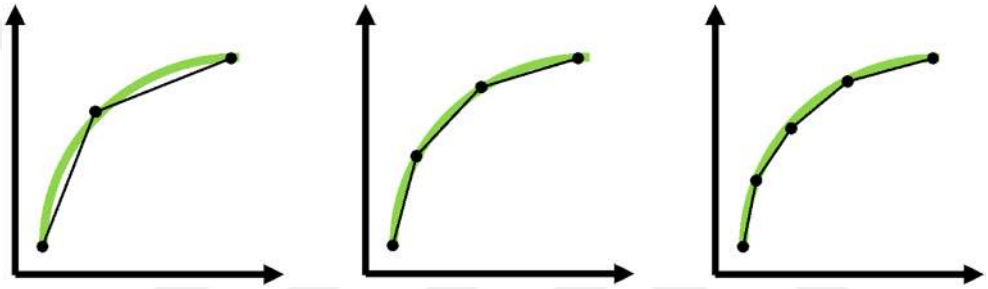


Figure 1.11 Fitting lines to an arc with different error thresholds.

Although the poor performance of line detection algorithm based on Hough transform (Luo and Zhao, 2022), the HT (Chen et al., 2021) is still popular algorithm in the literature. Nowadays, there are a lot of studies to improve the HT (Li et al., 2021; Du et al., 2021). Luo and Zhao improve the Hough transform by using parameter space block to detect the road lane lines. Li et al. use the improved Hough transform to detect the rail track edge. Du et al. improve the Hough transform by using Minimum Region Method to overcome the shortcoming of the HT. Chen et al. implement Hough transform in the encrypted domain for the line detection on encrypted images.

Recently, The LSD (Teplyakov et al., 2022) and the EDLines (Zhang et al., 2021) are very famous line segment detectors (Ossimitz and TaheriNejad, 2021; Xu et al., 2021; Abdellali et al., 2021). The EDLines and the LSD have very similar results but the EDLines is faster than the LSD. Teplyakov et al. replace the first step

of the LSD with a lightweight convolutional neural networks for a fast and accurate line segment detection. Zhang et al. use the improved EDLines algorithm for detection of the dividing line between the sky area and the water surface area. Ossimitz and TaheriNejad improve the LSD algorithm by using a lookup table instead of calculating the logarithm values. Xu et al. introduce a line segment detector based on a multi-scale encoder/decoder transformer structure. Abdellali et al. present learnable line segment detector and descriptor for wide baseline line matching.

1.3. Research Objectives and Contributions

The main objectives of the thesis are described as follows:

- To develop an ultrafast pattern based line detector for real time image processing.
- To obtain efficient cost memory by using 4 × 4 pixel line patterns for a 64 KB look-up table.
- To offer 3 modes to fit different number of lines to arcs for graphical user interface

In this thesis, a novel fast line detector is accomplished and much of the work in this thesis is an extension on the work presented in the following publications:

- Yilmaz, I. C., and Baykal, I. C., 2022. Ultrafast line detector. Journal of Electronic Imaging, 31/4, 31, 043019-1/15, SPIE and IS&T (https://doi.org/10.1117/1.JEI.31.4.043019).
- Afser, H., and Yilmaz, I. C., 2021. Kararli hipotez testi tabanli interaktif çoklu görüntü kesimleme. 1. International Dicle Scientific Research and Innovation Congress, 168-174.

- Baykal, I. C., and Yilmaz, I. C., 2017. An Extremely Fast Pattern Based Line Detector. 13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)., 369-376, 10.1109/ICCP.2017.8117032.
- Daricik F., and Yilmaz, I. C., 2016. A Novel Method to Determine Interlaminar Fracture Characteristics of Laminated Composites. International Conference on Material Science and Technology in Cappadocia, 377-381.

1.4. The Organization of the Thesis

This thesis is organized as follows:

Chapter 1: Introduction

A brief introduction about line detectors in the image processing is presented in the first chapter. Theoretical background, main motivation, literature review, research objective, original contributions and organization of the thesis are stated in the present chapter.

Chapter 2: Hough, LSD and EDLines

In this chapter, the famous Hough, LSD and EDLines line detector algorithms in the literature, are briefly explained.

Chapter 3: Proposed Line Detector Methods

In this chapter, the proposed line detector method with the look-up tables for 110 4 × 4 pixel line patterns is rigorously explained.

Chapter 4: Results and Speed Tests

Chapter 4 presents detailed the speed test results of the proposed line detector algorithm given in Chapter 3. The test results are expressed graphically and tabulated in terms of speed performance.

Chapter 5: Conclusions and Future Works

Chapter 5 gives some concluding remarks and future perspectives belonging to the thesis.

Finally, the bibliography and the biography of the author are presented.

1.5. Conclusions

In this part of the thesis, basic features of the research including the research goal, problem definition, main motivations and original contributions are presented.

.

2. HOUGH, LSD AND EDLINES

2.1. Hough

In Hough space (Hough, 1962), HT converts every edge pixel into sinusoids and proves that the sinusoids of each pixel belonging to a line always pass through one single mathematician point in Figure 2.1. The intersection points of these sinusoids give us the equations for the lines. On the other hand, in the discrete space, there are always 8 pixels surrounding a single pixel in Figure 2.2. Moreover, those 8 pixels are surrounded by neighbour 16 pixels and create flat peaks instead of a spike in Figure 2.3.

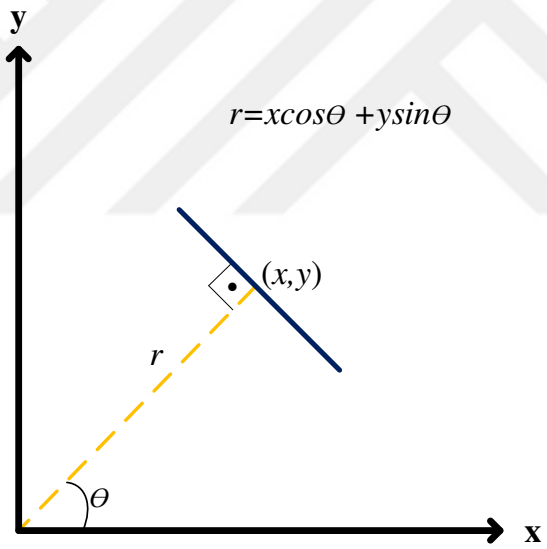


Figure 2.1 HT line parametrization.

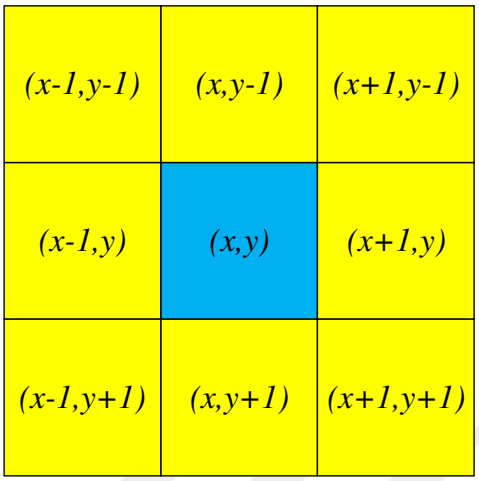


Figure 2.2 Yellow 8 pixels surrounding a single blue pixel.

Figure 2.3 shows demonstrating the intersection of seven sinusoids. Each gray sinusoid represents one pixel of the same line and there are also dark sinusoids which belong to pixels not connected to the line but have sinusoids coincidentally passing by the intersection point. The intersection area creates a flat peak when superimposed, instead of a sharp point even when the line is seven pixels long.

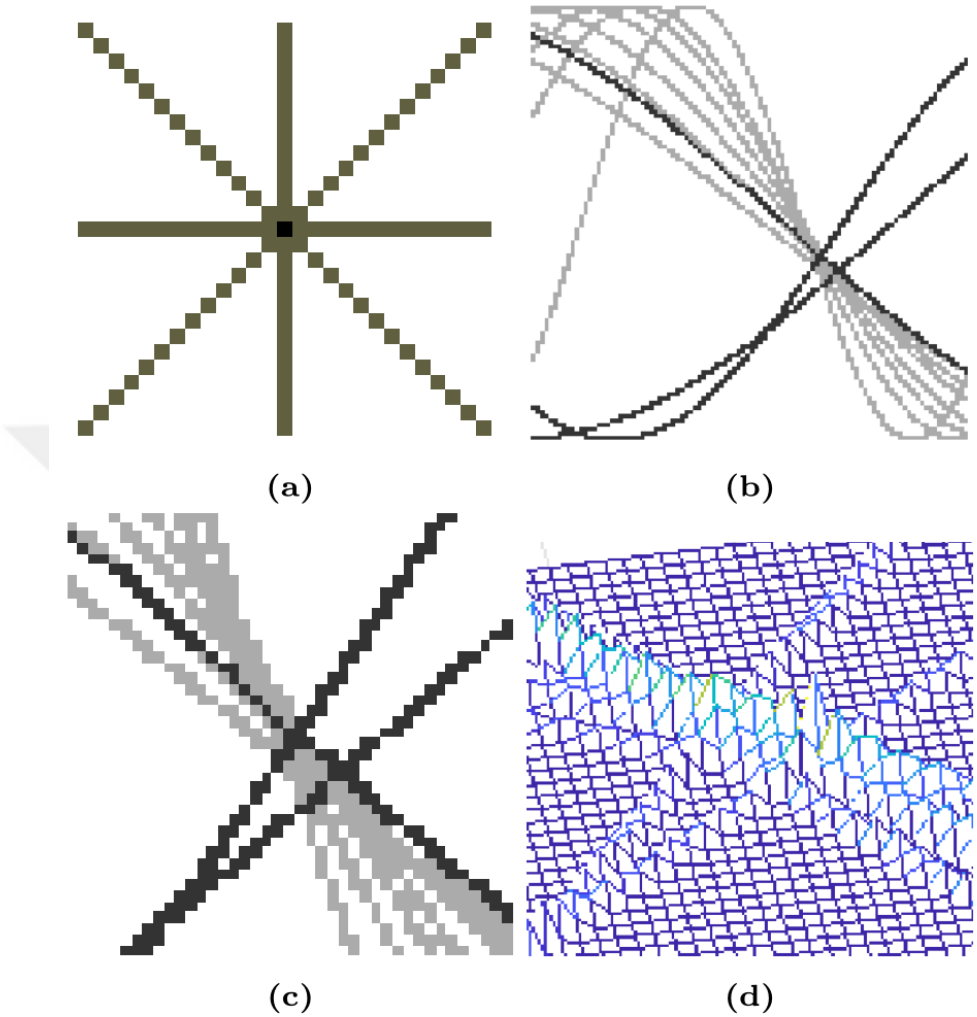
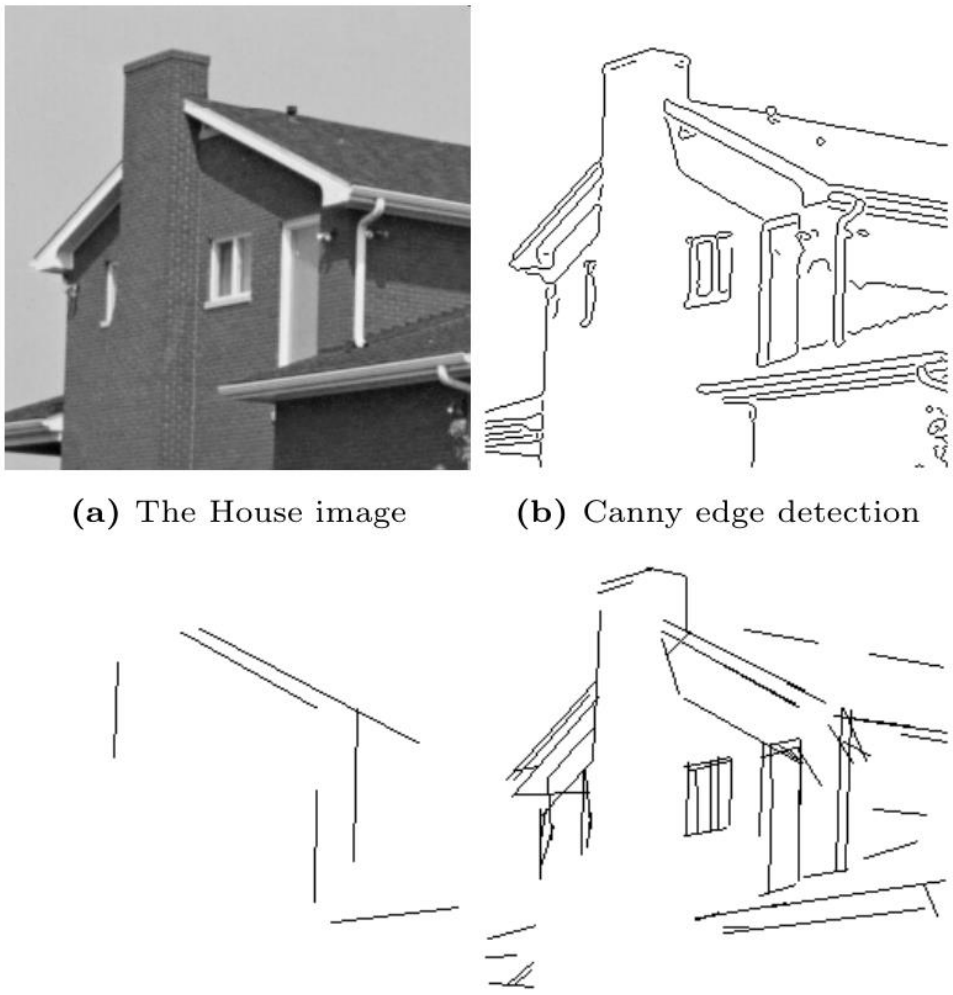


Figure 2.3 (a) Intersection of 4 lines. (b) Intersection of sinusoids belonging to a 7 pixel long line (gray) and 3 random pixels (dark) in Hough domain. (c) Magnified intersection point showing that sinusoids do not intersect at a single point in the discrete space. (d) Intersection area in 3-D.

Figure 2.4 shows the House image at 256 × 256 pixel resolution and the poor results of the Probabilistic HT.



(a) The House image **(b)** Canny edge detection

(c) PHT (PI/180, 50, 50, 10) **(d)** PHT (PI/180, 13, 7, 10)

Figure 2.4. (a) The House image (Matas et al., 2000). (b) Canny edge detection. (c) PHT with arguments (PI/180, 50, 50, 10). (d) PHT with arguments (PI/180, 13, 7, 10).

2.2. LSD

The LSD algorithm (Von Gioi et al, 2008) groups adjacent pixels with similar gradient angle for line detection. Each edge pixel has the gradient magnitude and gradient angle calculated with a 2 × 2 kernel matrix. LSD (Ossimitz, 2021) uses scaling (blurring) by a factor of 0.8 to reduce image noise as a first step. By using thresholding technique, LSD eliminates edge pixels with weak gradient magnitude. LSD uses the region (rectangular) method with a contario approach for line segment detection. Figure 2.5 shows the block diagram of the LSD algorithm and Figure 2.6 shows the result of the LSD for the input House image.

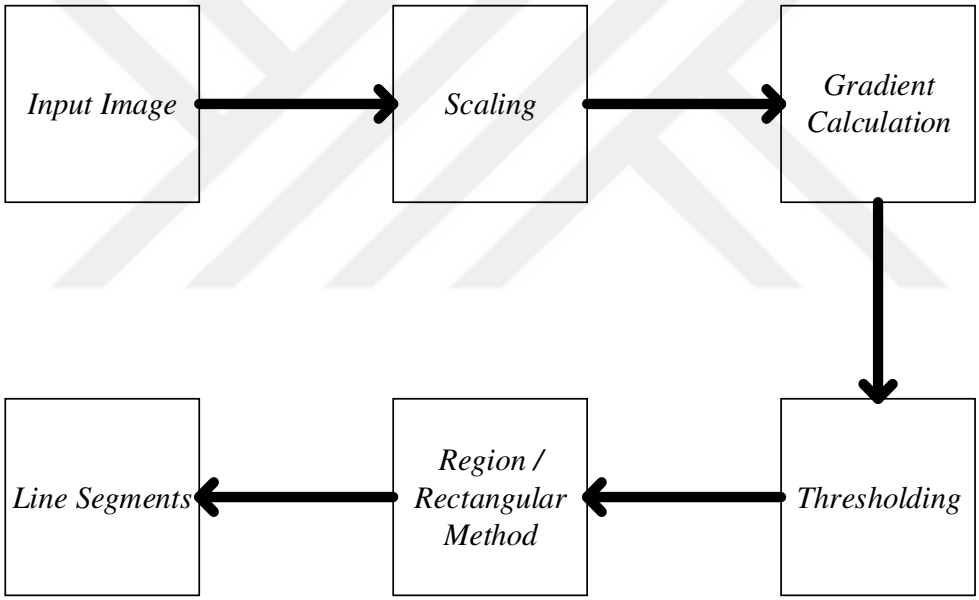


Figure 2.5 The block diagram of the LSD algorithm.

Figure 2.6 The result of the LSD for the input House image.

2.3. EDLines

The EDLines algorithm (Akinlar and Topal, 2013) is based on improving the LSD by using edge drawing (ED) (Topal and Akinlar, 2012). The ED algorithm (Ossimitz, 2021) is an edge detection algorithm and uses a smoothing filter for noise reduction as a first step. The EDLines has the same kernel matrix as the LSD to calculate the gradient magnitude and gradient direction. After the gradient magnitude map, the connected peak pixels, anchors, are linked as a pixel chain to draw edge pixels. Finally, EDLines fits lines segments onto each pixel chain by using the least squares fitting (LSF) algorithm and validates the line segments by using the number of false alarms (NFA). Figure 2.7 shows the block diagram of the EDLines algorithm and Figure 2.8 shows the result of the EDLines for the input House image.

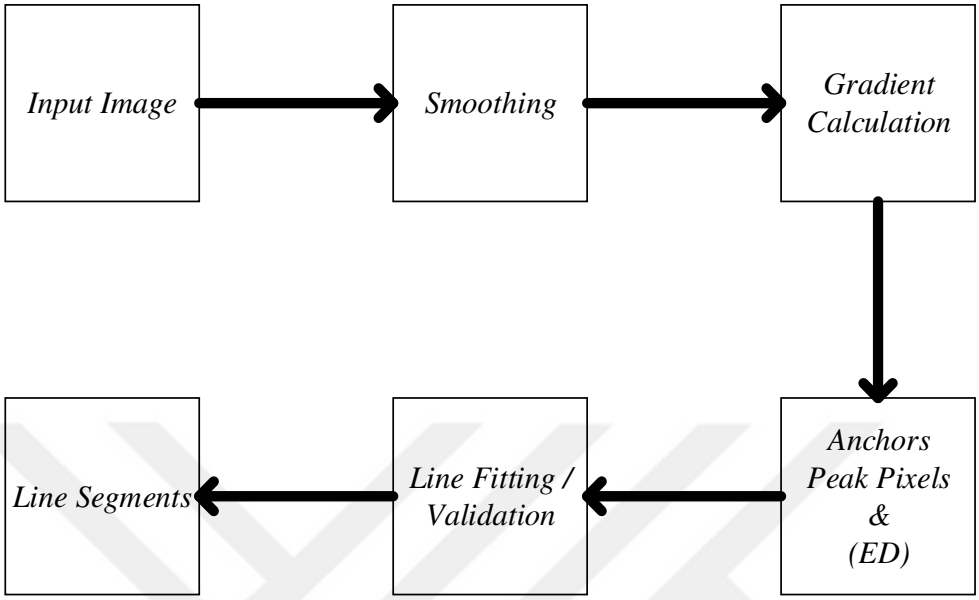


Figure 2.7 The block diagram of the EDLines algorithm.

Figure 2.8 The result of the EDLines for the input House image.

3. PROPOSED LINE DETECTOR METHODS

3.1. Introduction

This section presents the proposed line detector method with the look-up tables for 110 4 × 4 pixel line patterns.

3.2. Ultrafast line Detector

The block diagram of the proposed line detector algorithm is shown in Figure 3.1. Firstly, to find the edges in an image, an edge detector with thinning is applied to an image and then the binary edge image is divided to 4 × 4 pixel windows. In this thesis, the Canny methods and the EDLines edge detector are used for the binary edge image. 110 different 4 × 4 pixel patterns can generate any straight line with an arbitrary angle. These patterns are shown in Figure 3.2. Too many artificial images containing numerous lines with different slopes are scanned to obtain these patterns. All 110 4 × 4 pixel edge line patterns have 8 different slope angle and the tip points named start and end. Any 4 × 4 pixel line pattern is essentially sixteen on/off bits so that each line pattern has a 16-bit unsigned integer value between 0 and 65,535 and is addressed to a 64 KB look-up table in Table 3.1

The first step of the proposed algorithm is that recognizing these patterns in a binary edged image and eliminating edge pixels that cannot be a part of a straight line. Eliminating the unrecognized edge pixels makes this algorithm incredibly fast. An ingeniously simple method is used to recognize 4 × 4 pixel patterns; converting 4 × 4 edge pixels into a 16-bit unsigned integer and checking that integer is addressed whether or not in look-up table.

The conversion weight for each pixel inside the 4 × 4 pattern is shown in Figure 3.3 and 22.5 degree apart eight possible slopes and their numeric labels are shown in Figure 3.4.

The main advantage of using the 4 × 4 pixel pattern is that a 64 KB unsigned char array is enough to implement look-up table. If 3 × 3 pixel patterns were used, there would have been far less patterns than 110 and 3 × 3 pixel patterns would have been too simple to work. On the other hand, if 5 × 5 pixel patterns were used, look-up table would have required 32 MB instead of 64 KB because of more than 110 patterns.

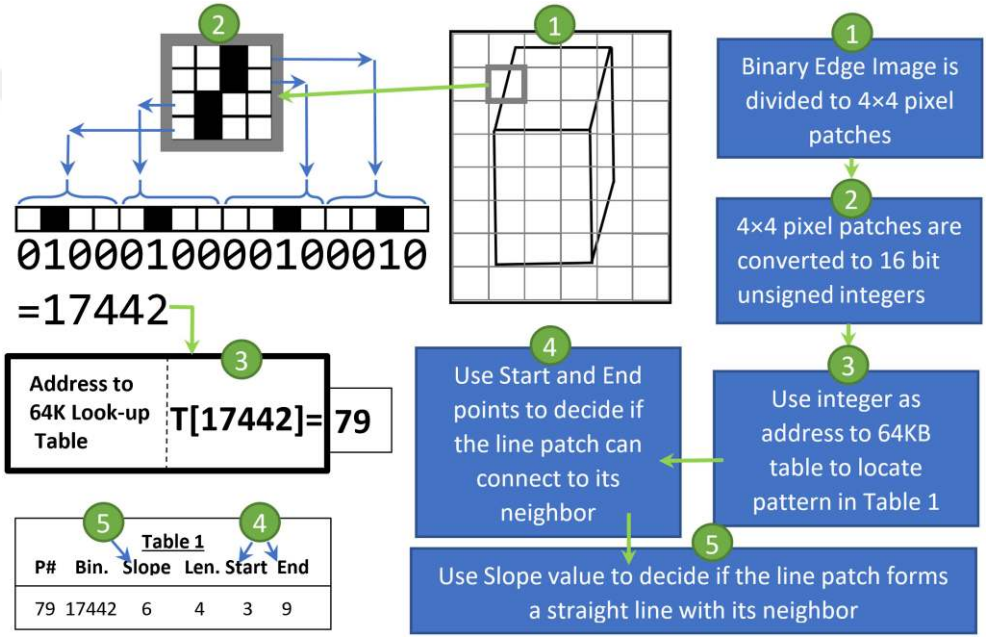


Figure 3.1 The block diagram of the proposed algorithm.

Table 3.1 contains the values of those 16-bit integers along with their slope labels in Figure 3.2, length in terms of pixels and connection point information.

According to the neighbourhood rule, these patterns can only be connected as a line if their slopes and connection points are compatible. The neighbourhood rule is ensured by two other small look-up tables.

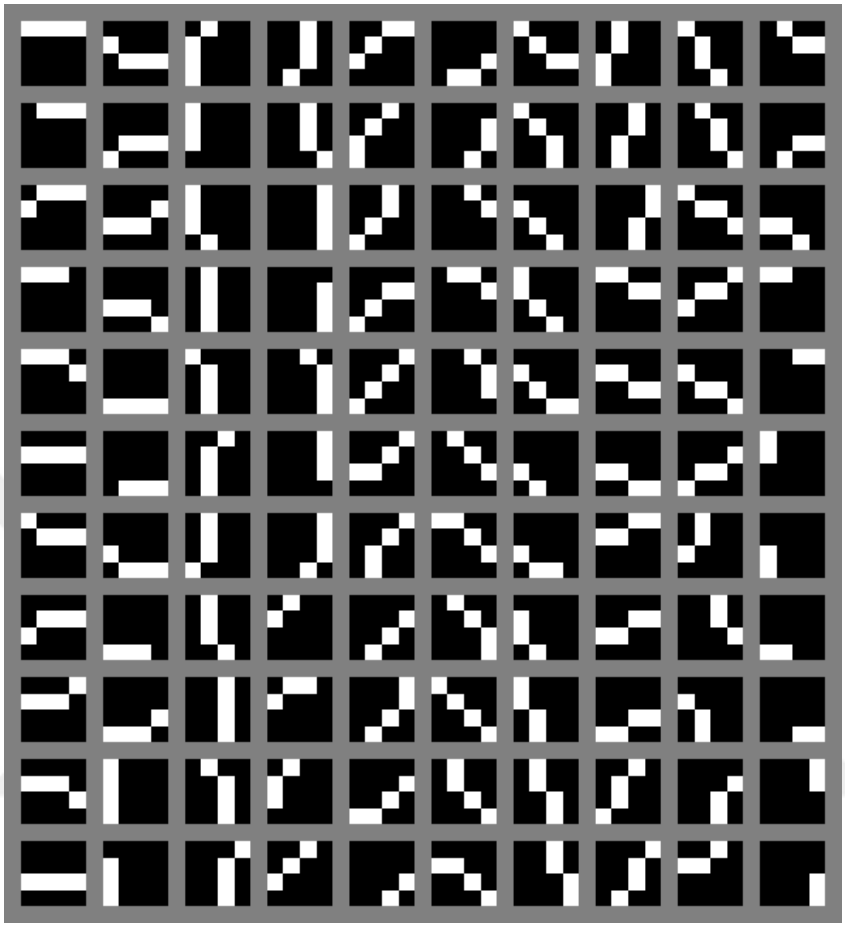


Figure 3.2 110 different 4 × 4 patterns that a line can create.

Figure 3.5 shows a simple 240 × 204 pixel resolution image of boxes as a input for the proposed algorithm. Figure 3.6 demonstrates an example of recognition of 4 × 4 edge patterns. Firstly, the Canny edge detector is applied to the image shown in Figure 3.5. Secondly, the image is divided to 4 × 4 windows for converting each window to 16-bit unsigned integer. Finally, this integer is used as an address to a 64 KB look-up table. If this value is one of the 110 patterns shown in Figure 3.2 or one of the entries in Table 3.1, then that 4 × 4 window is painted in light gray which means recognized in Figure 3.6. If that integer is not one of those 110 entries, then the look-up table is zero at that address and the 4 × 4 window is painted in dark blue

which means unrecognized in Figure 3.6. The dark blue windows are generally noticeable at the tip points or at the intersection points of the lines.

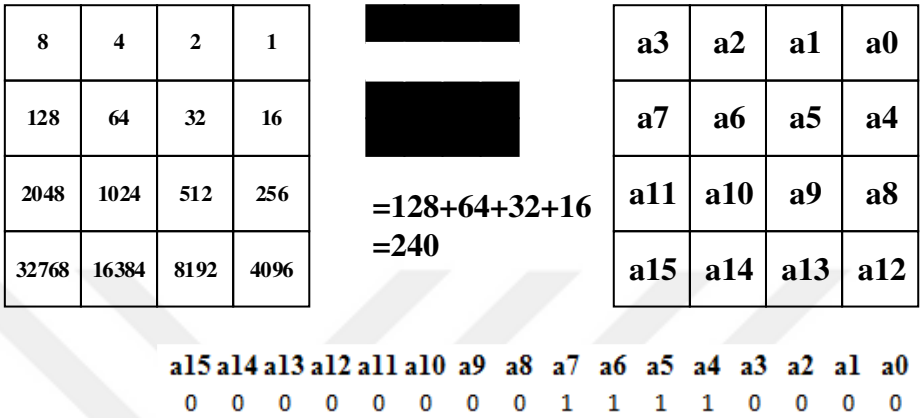


Figure 3.3 Binary conversion weights and the conversion of a 4x4 pixel line pattern into a 16 bit unsigned integer.

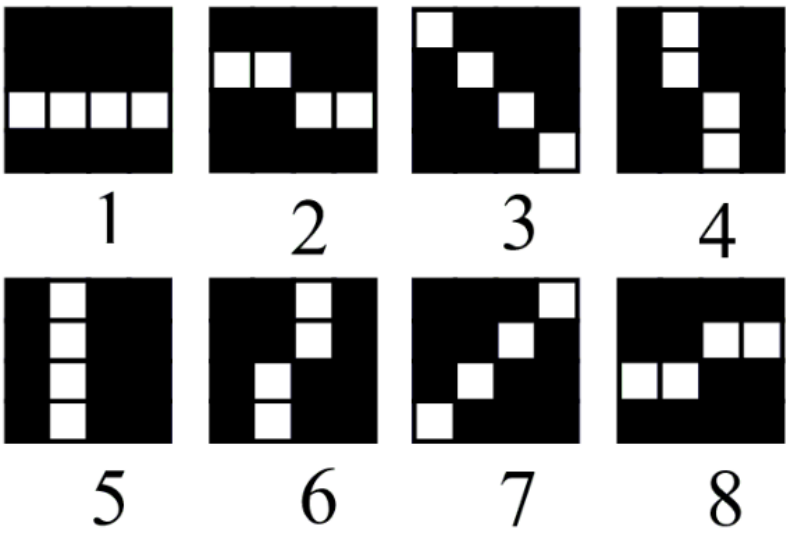


Figure 3.4 22.5 degree apart eight possible slopes and their numeric labels.

Table 3.1 Binary values of line patterns, their slope angle, length and end points

P#	Bin.	Slope	Len.	Start	End	P#	Bin.	Slope	Len.	Start	End
1	15	1	4	1	4	56	24832	8	3	6	9
2	7	2	3	2	4	57	8464	6	3	5	8
3	135	8	4	12	4	58	18	3	2	3	5
4	14	8	3	1	3	59	22	2	3	2	5
5	30	2	4	1	5	60	274	4	3	3	6
6	240	1	4	12	5	61	292	3	3	2	6
7	120	2	1	5	4	62	300	2	4	1	6
8	2160	8	4	11	5	63	4388	4	4	2	7
9	225	8	4	12	4	64	4680	3	4	1	7
10	480	2	4	12	6	65	4676	4	4	2	7
11	3840	1	4	11	6	66	4800	2	4	12	7
12	1920	2	4	12	6	67	840	2	4	1	6
13	34560	8	4	10	6	68	8776	4	4	1	8
14	3600	8	4	11	5	69	9344	3	3	12	8
15	7680	2	4	11	7	70	9352	4	4	1	8
16	61440	1	4	10	7	71	13440	2	4	12	7
17	28672	8	3	9	7	72	18432	3	2	11	9
18	30720	2	4	11	7	73	26624	2	3	11	8
19	57344	2	3	10	8	74	18560	4	3	12	9
20	57600	8	4	10	6	75	17544	4	4	1	9
21	34952	5	4	1	10	76	8772	4	4	2	8
22	34944	4	3	12	10	77	4386	4	4	3	7
23	34948	6	4	2	10	78	34884	6	4	2	10
24	2184	6	3	1	11	79	17442	6	4	3	9
25	18568	4	4	1	9	80	8721	6	4	4	8
26	17476	5	4	2	9	81	60	2	4	1	5
27	17480	4	4	1	9	82	960	2	4	12	6
28	17474	6	4	3	9	83	15360	2	4	11	7
29	33860	6	4	2	10	84	195	8	4	12	4
30	9284	4	4	2	8	85	3120	8	4	11	5
31	8738	5	4	3	8	86	49920	8	4	10	6
32	8740	4	4	2	8	87	33858	6	4	3	10
33	8737	6	4	4	8	88	16929	6	4	4	9
34	16930	6	4	3	9	89	9288	4	4	1	8
35	4642	4	4	3	7	90	4644	4	4	2	7
36	4369	5	4	4	7	91	360	2	4	1	6
37	4368	6	3	5	7	92	2145	8	4	11	4
38	4370	4	4	3	7	93	5760	2	4	12	7
39	273	4	3	4	6	94	34320	8	4	11	4
40	8465	6	4	4	8	95	2116	6	3	2	11
41	132	7	2	2	12	96	290	4	3	3	6
42	134	8	3	12	3	97	17536	4	3	12	9
43	2180	6	3	2	11	98	8720	6	3	5	8
44	2114	7	3	3	11	99	194	8	3	3	12
45	2115	8	4	11	4	100	52	2	3	2	5
46	34882	6	4	10	3	101	17152	8	3	6	9
47	33826	6	4	10	3	102	11264	2	3	11	8
48	33840	8	4	10	5	103	12	8	2	1	2
49	33825	7	4	10	4	104	3	2	2	3	4
50	3105	8	4	11	4	105	136	6	2	1	12
51	17441	6	4	9	4	106	34816	4	2	11	10
52	16912	7	3	9	5	107	49152	2	2	10	9
53	16913	6	4	9	4	108	12288	8	2	8	7
54	49680	8	4	10	5	109	17	4	2	4	5
55	8448	7	2	8	6	110	4352	6	2	6	7

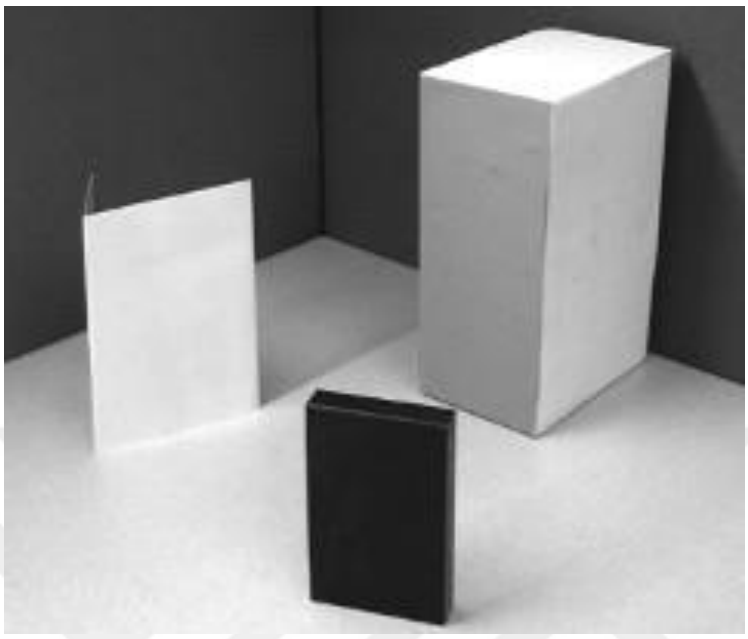


Figure 3.5 A simple image of boxes.

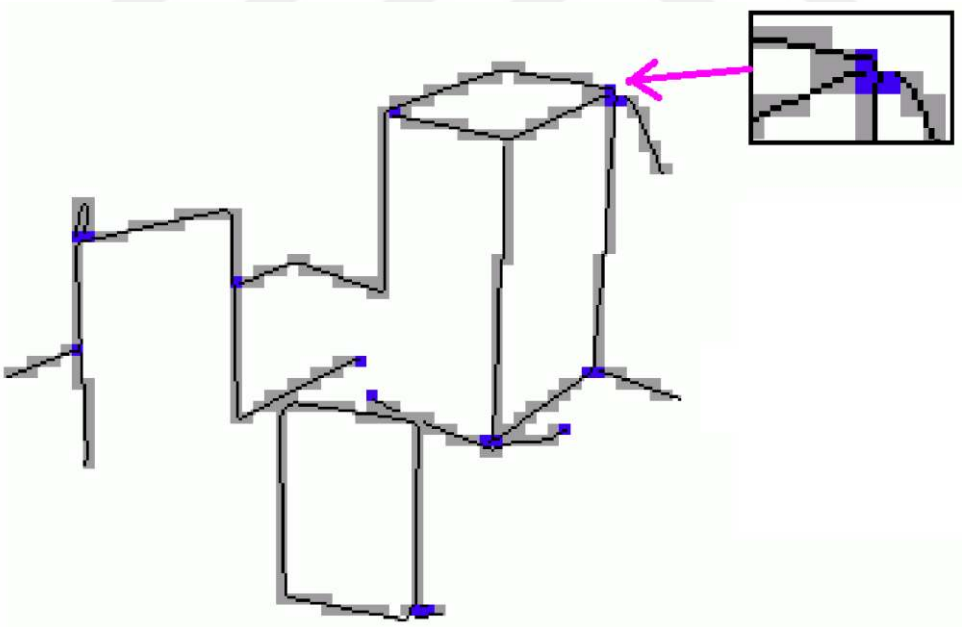


Figure 3.6 The result of the recognition of 4 × 4 pixel patterns shown in Figure 3.5.

Figure 3.7 shows the result of the Canny ([0.05,0.1],1.2) edge detection for the input House image in Figure 2.4. Figure 3.8 shows the unrecognized 4 × 4 pixel line patterns in red windows and recognized patterns in grey.

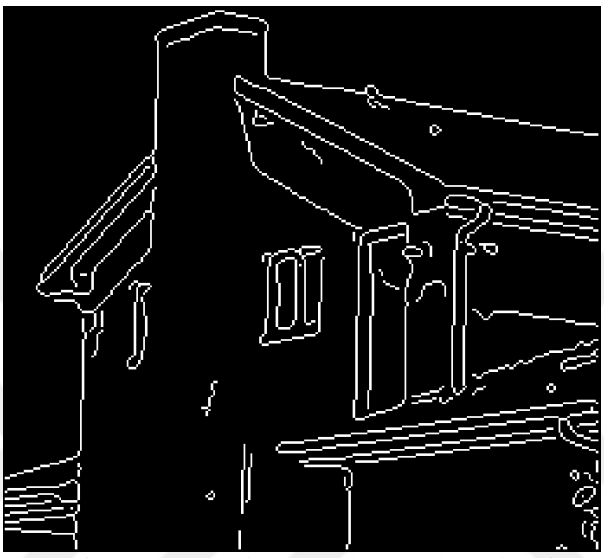


Figure 3.7 The result of Canny ([0.05,0.1],1.2) edge detection for the House image.

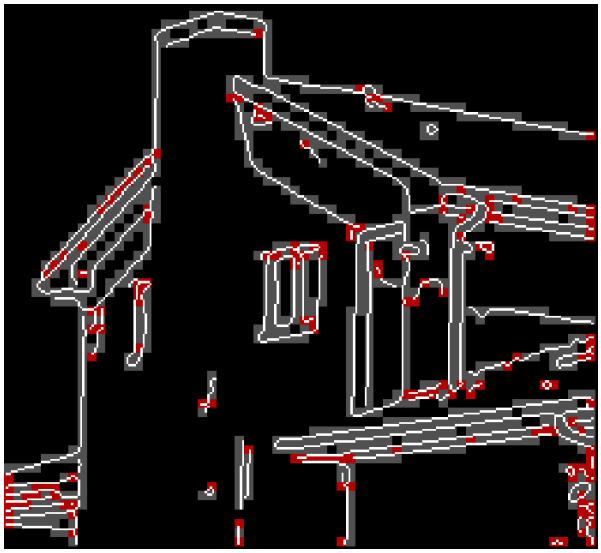


Figure 3.8 Unrecognized edge pixels in red 4 × 4 windows and recognized in grey.

Figure 3.9 shows the zoomed regions of Figure 3.8. According to the zoomed regions, the edge pixels close to each other <5 pixels are unrecognized generally.

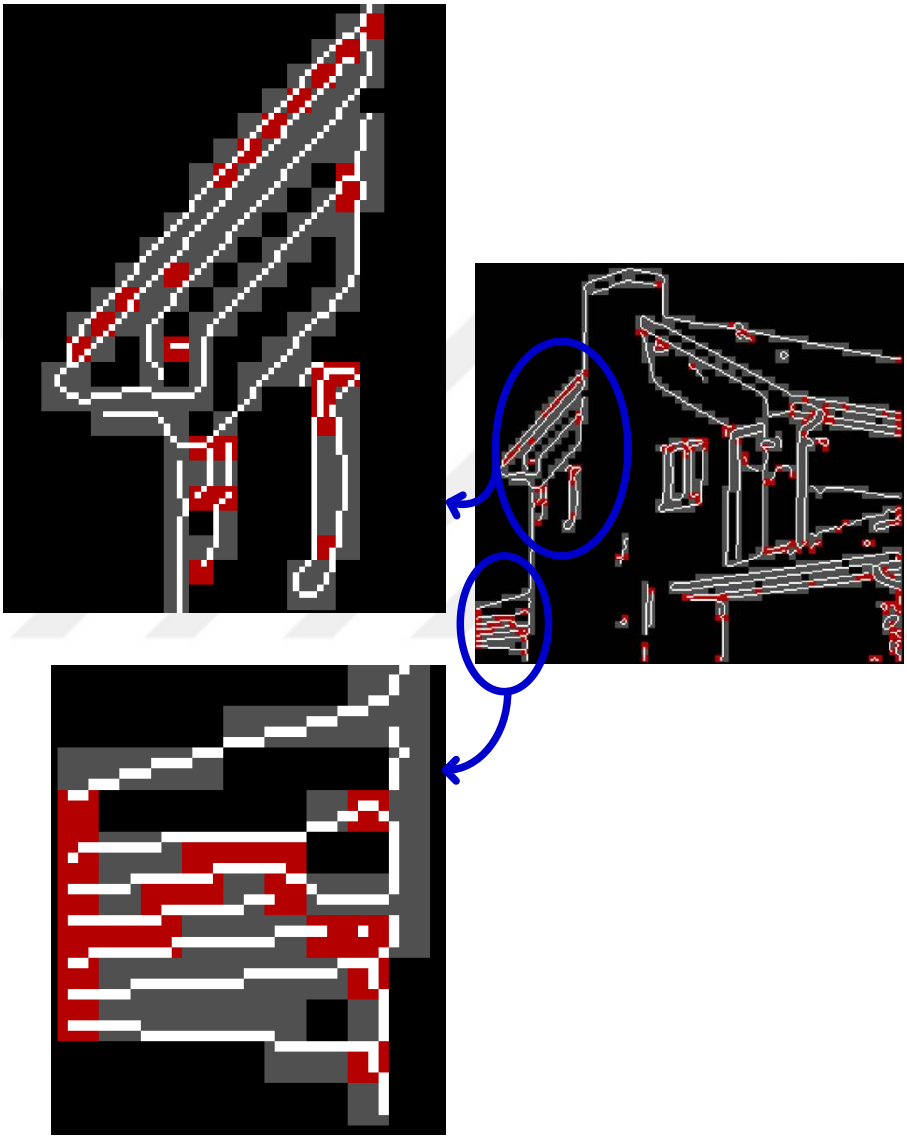


Figure 3.9 Unrecognized pixels close to each other <5 pixels in the zoomed regions.

Figure 3.10 shows the slope colors of the recognized 4 × 4 pixel line patterns; 1=Gold, 2=Green, 3=Blue, 4=Magenta, 5=Yellow, 6=Cyan, 7=Dark Green 8=Dark Blue for the House image.

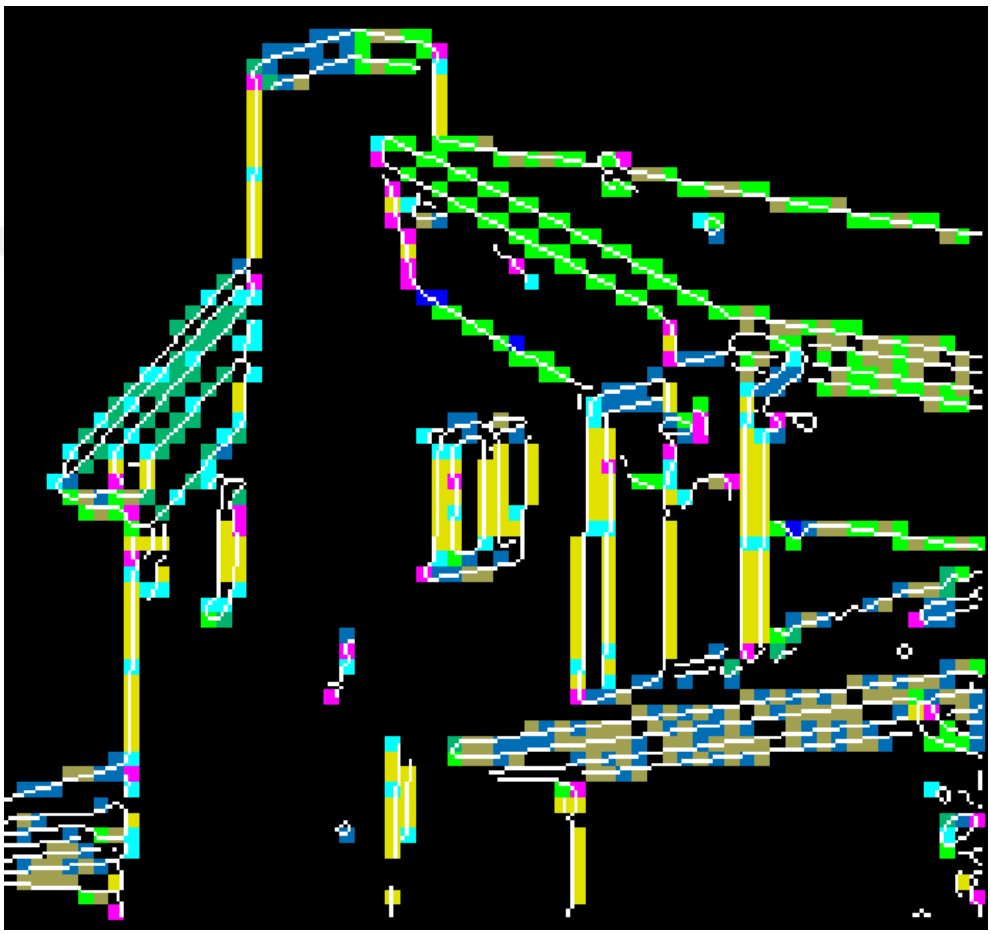


Figure 3.10 Slopes of the recognized 4 × 4 pixel line patterns; 1=Gold, 2=Green, 3=Blue, 4=Magenta, 5=Yellow, 6=Cyan, 7=Dark Green 8=Dark Blue.

Figure 3.11 shows the result of this process when applied to the edge detected tree picture shown in Figure 1.8. The tiny dark gray squares are the correctly recognized 4 × 4 edge patterns and these recognized patterns form lines as seen in Figure 1.9. The pink squares indicate the patterns that can not be line patterns and

these pink patches are eliminated as seen in Figure 1.9. These eliminated patterns make the proposed algorithm extremely fast.

Figure 3.11 The unrecognized 4 × 4 pixel edge patterns are painted in pink and eliminated.

3.3. Segmentation of 4×4 Patterns

The first step of segmentation of 4 × 4 patterns is checking the neighboring 4×4 edge patterns in terms of their slope. In Table 3.1, every line pattern has a slope

value labelled from 1 to 8 based on their angle as seen in Figure 3.4. Actually, these labels represent slopes that are integer multiple of 22.5 degree.

A 4 × 4 pixel line pattern window is surrounded by eight other line patterns as shown in Figure 3.12. However, we do not need look tables for every eight locations surrounding our target pattern. For example, if the processing starts from the top left corner of the image and proceeds left to right and top to bottom, only four neighbors (left, up-left, up and up-right) are considered because those are the only neighbors that have already been processed. The rest of the neighbors are redundant so that the proposed algorithm is ultra-fast. Therefore, totally four (the top three neighbors plus the left neighbor) neighbors are processed by the look-up tables. In Figure 3.13, the top three neighbors and the left neighbor are painted in light grey while the target (center) pattern is shown using darker grey.

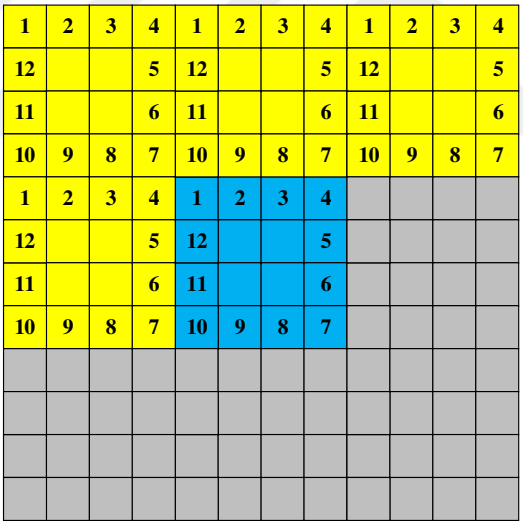


Figure 3.12 4 × 4 pixel blue target pattern needs only four yellow neighbors for segmentation. Grey neighbors are not used.

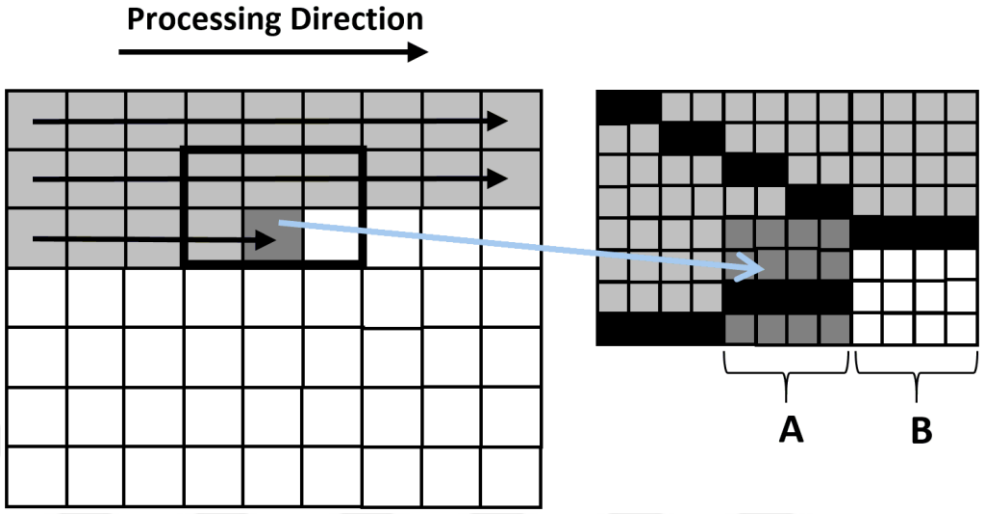


Figure 3.13 Processing direction and the zoomed target pattern with its neighbors.

The relative position and the slope label of a neighbor are very important while deciding which patterns can connect for segmentation. For example, if the 4 × 4 pixel line pattern is a horizontal line pattern with slope label 1 such as shown in Figure 3.13, then it cannot connect to any line patterns that are exactly up or down it and it can only connect to patterns that have slope label of 1, 2, or 8 if those patterns are on the right or left side.

In Figure 3.14, the 4 × 4 pixel line pattern has the tip points 11 and 6 so that it can only segment the neighbor pattern with the tip points 10,11 and 12 for the tip point 11 and 5, 6 and 7 for the tip point 6. Therefore, a small look-up table is used for each location defining which slopes can connect to each other in Table 3.2 and another small look-up table is used for connectivity rules based on the connection points of neighboring patterns in Table 3.3.

According to the Table 3.3, pattern “A” with tip points (11,6) and pattern “B” tip points (1,4) cannot be segmented although their slopes are same.

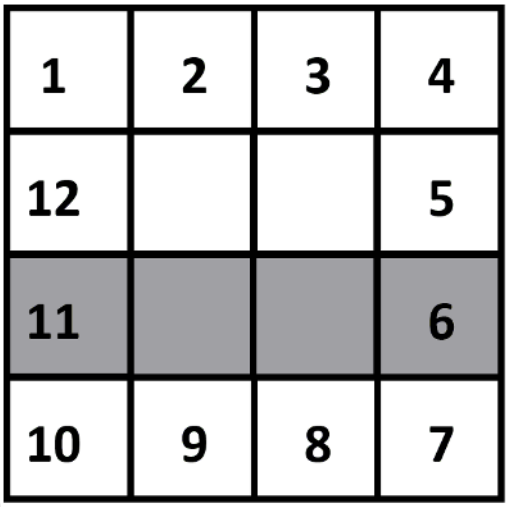


Figure 3.14 The connection points of the 4 × 4 target pattern in Figure 3.13.

According to the Table 3.2, the 4 × 4 pixel line pattern with slope 1 can only be connected to patterns with slope 1, 2 or 8. Pattern with slope 2 can only be connected to patterns with slope 1, 2 or 3. Pattern with slope 3 can only be connected to patterns with slope 2, 3 or 4. Pattern with slope 4 can only be connected to patterns with slope 3, 4 or 5. Pattern with slope 5 can only be connected to patterns with slope 4, 5 or 6. Pattern with slope 6 can only be connected to patterns with slope 5, 6 or 7. Pattern with slope 7 can only be connected to patterns with slope 6, 7 or 8. Pattern with slope 8 can only be connected to patterns with slope 7, 8 or 1.

Table 3.2 Slope connectivity rules for neighboring patterns (modes 0 and 1)

Slope	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	1	1	0	0	0	0	0	1
2	1	1	1	0	0	0	0	0
3	0	1	1	1	0	0	0	0
4	0	0	1	1	1	0	0	0
5	0	0	0	1	1	1	0	0
6	0	0	0	0	1	1	1	0
7	0	0	0	0	0	1	1	1
8	1	0	0	0	0	0	0	0

Table 3.3 Connectivity rules based on the connection points of neighboring patterns.

Target 4 × 4 pixel connection points	Up neighbor 4 × 4 pixel connection points				Left neighbor 4 × 4 pixel connection points			
	10	9	8	7	4	5	6	7
1	1	1	0	0	1	1	0	0
2	1	1	1	0	0	0	0	0
3	0	1	1	1	0	0	0	0
4	0	0	1	1	0	0	0	0
12	0	0	0	0	1	1	1	0
11	0	0	0	0	0	1	1	1
10	0	0	0	0	0	0	1	1

For example, Figure 3.15 shows 24 × 20 edge pixels and Figure 3.16 shows the binary conversion of Figure 3.15.

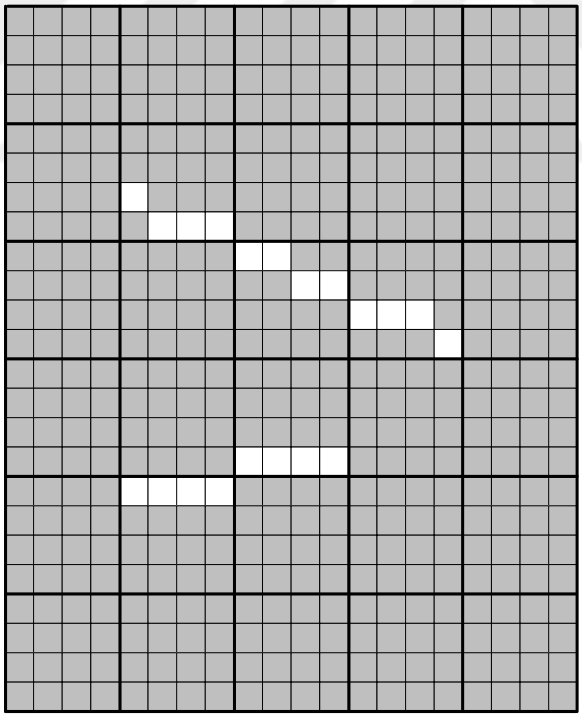


Figure 3.15 Example of 24 × 20 edge pixels.

0	0	0	0	0
0	30720	0	0	0
0	0	60	7680	0
0	0	61440	0	0
0	15	0	0	0
0	0	0	0	0

Binary	P#	Slope	Start	End
30720	18	2	11	7
60	81	2	1	5
7680	15	2	11	7
61440	16	1	10	7
15	1	1	1	4

Figure 3.16 Conversion 6 × 5 binary matrix of Figure 3.15.

In Figure 3.16, there are 5 recognized line patterns and their segmentation steps;

- Pattern 18 (Binary value 30720) does not connect any left, up-left, up and up-right neighbors so that its segmentation number is 1.
- Pattern 81 (Binary value 60) connects to up-left pattern 18 according to the slope and connectivity rules and it takes the segmentation number of pattern 18 so that its segmentation number is 1.
- Pattern 15 (Binary value 7680) connects to left pattern 81 and it takes the segmentation number of pattern 81 so that its segmentation number is 1.
- Pattern 16 (Binary value 61440) does not connect any left, up-left, up and up-right neighbors according to the slope and connectivity rules so that its segmentation number is 2.

- Pattern 1 (Binary value 15) connects to up-right pattern 16 according to the slope and connectivity rules and it takes the segmentation number of pattern 16 so that its segmentation number is 2.

Figure 3.17 shows the segmentation result of Figure 3.15. In Figure 3.17, yellow patterns have segmentation number 1 and blue patterns have segmentation number 2. Actually, there are 20 edge pixels in Figure 3.15 and segmenting only five recognized line patters instead of 20 edge pixels makes the proposed algorithm ultrafast line detector.

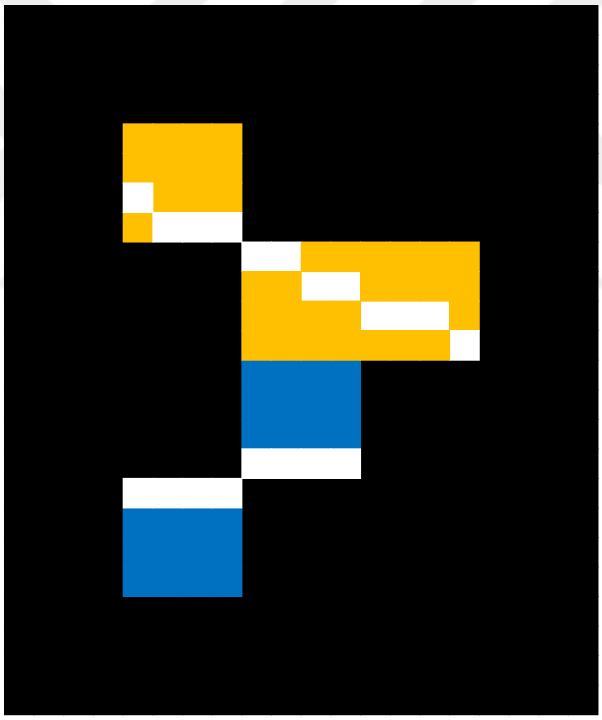


Figure 3.17 The segmentation result of Figure 3.15. Yellow patterns have segmentation number 1 and blues 2.

Figure 3.18 shows the detected lines of Figure 3.15. In Figure 3.18, yellow cross signs show the start point of the lines and blue cross signs show the end point

of the lines. Figure 3.19 shows the detected lines on the edge pixels in Figure 3.15. According to the Figure 3.19, the proposed algorithm does not need any validation step.

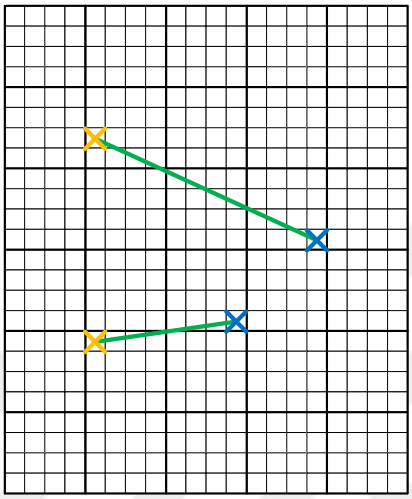


Figure 3.18 Yellow cross signs are start and blues are end points of the detected lines.

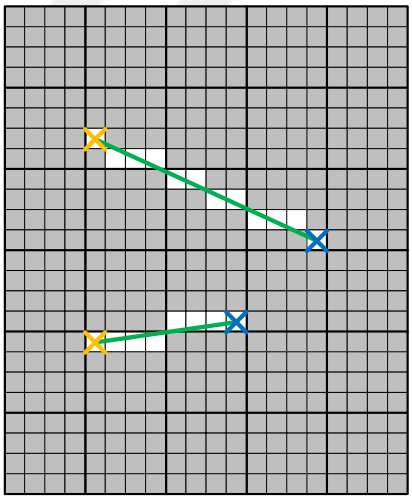


Figure 3.19 The detected lines on the edge pixels does not need any validation step.

Figure 3.20 shows the proposed segmentation result of the recognized 4 × 4 pixel line patterns in Figure 3.6. In Figure 3.20, the recognized 4 × 4 patterns that

belong to the same segment are painted using the same color. After the segmentation, there is left to do is find the tip points of those segments. These tip points are marked with tiny red and green crosses in Figure 3.21. As the final step of the algorithm is fitting the lines to the detected tip points. Figure 3.21 shows the corresponding lines to the segmentation result in Figure 3.20.

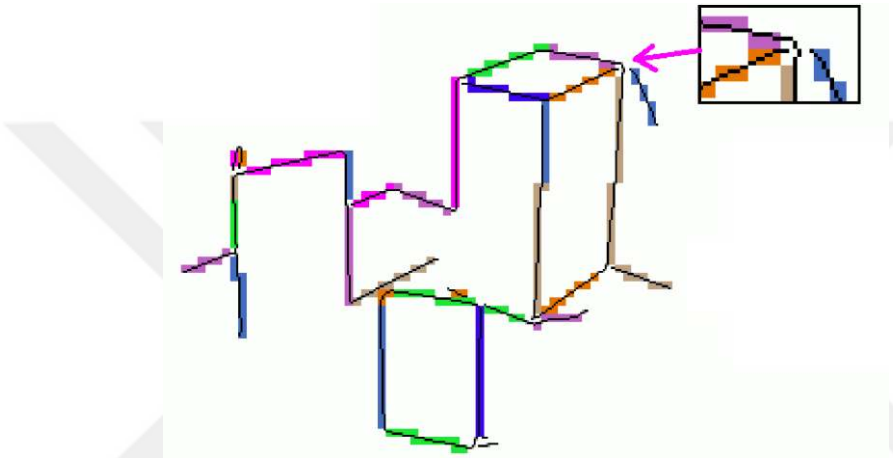


Figure 3.20 The result of the recognized patterns segmentation in Figure 3.6.

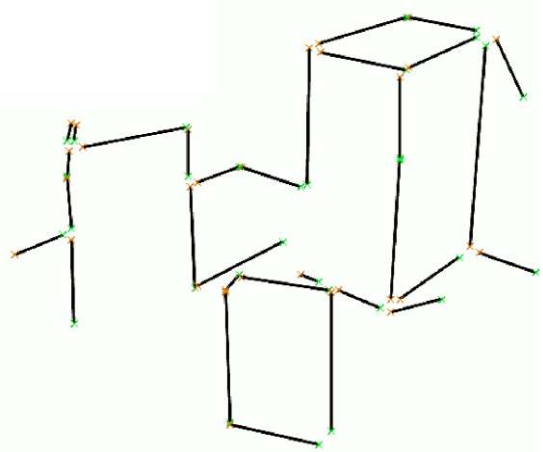


Figure 3.21 The lines corresponding to segments with start red and end green cross.

Figure 3.22 shows the segmentation result of the recognized 4 × 4 pixel line patterns in Figure 3.8. In Figure 3.22, the segmentation colors are defined by the segmentation numbers in modulus 9; 1 (mod 9) = Gold, 2 (mod 9) = Green, 3 (mod 9) = Blue, 4 (mod 9) = Yellow, 5 (mod 9) = Pink, 6 (mod 9) = Light Blue, 7 (mod 9) = Brown, 8 (mod 9) = Purple and 0 (mod 9) = Dark Green for the House image.

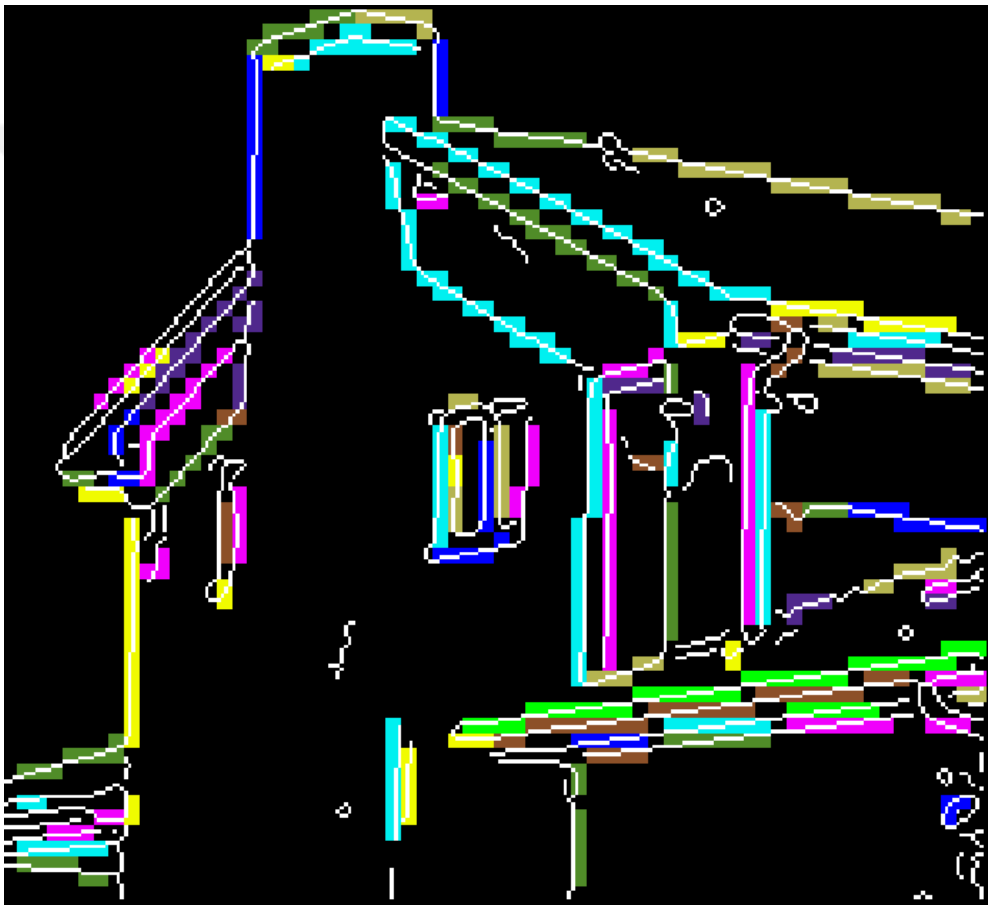


Figure 3.22 The result of the recognized patterns segmentation in Figure 3.8. Segmentation numbers in modulus 9; 1 (mod 9) = Gold, 2 (mod 9) = Green, 3 (mod 9) = Blue, 4 (mod 9) = Yellow, 5 (mod 9) = Pink, 6 (mod 9) = Light Blue, 7 (mod 9) = Brown, 8 (mod 9) = Purple and 0 (mod 9) = Dark Green.

Figure 3.23 shows the corresponding lines to the segmentation result in Figure 3.22.

Figure 3.23 The lines corresponding to segments in Figure 3.22. Yellow cross signs are the start point and blues are the end points of the lines.

Figure 3.24 shows the zoomed roof of the House image. The edge pixels of the roof look like an arc and are represented by two fitting lines.

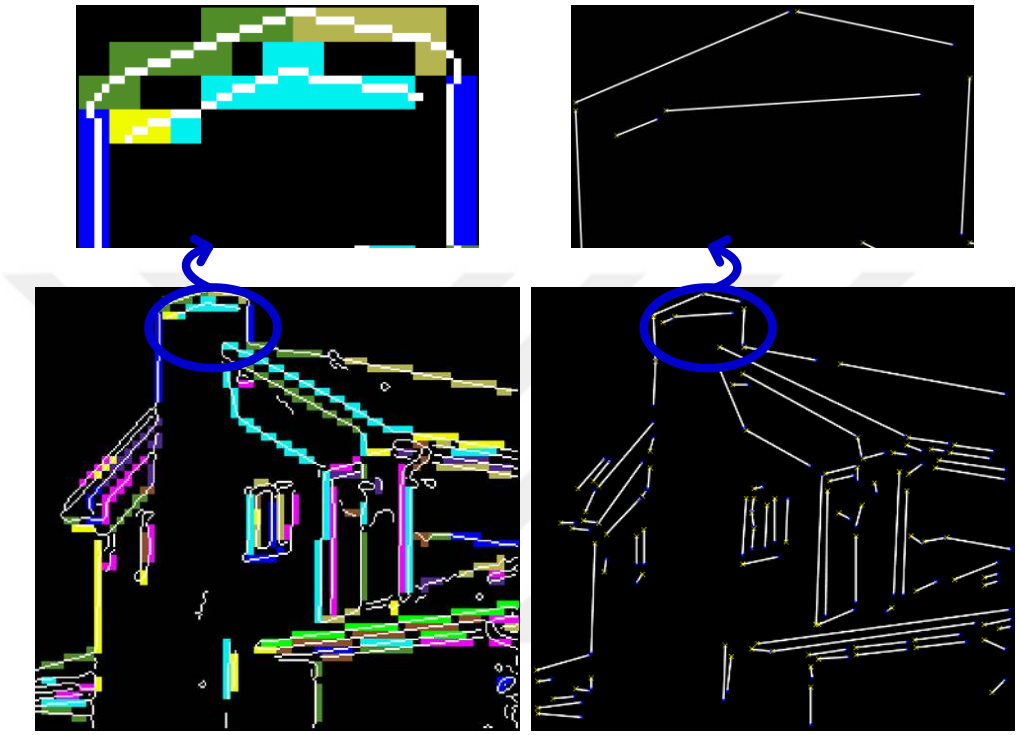


Figure 3.24 The edge pixels of the roof in zoomed region look like an arc represented by two fitting lines.

Fitting lines to an arc in Figure 1.11 is not a standard process and ideally requires the user to set an error threshold. If the error threshold is set high, then the arc is approximated by using less number of lines. On the other hand, if the threshold is too small, the arc is approximated more precisely with large number of lines. Because there is no “correct” or “golden” value for the error threshold for the line detectors, the correct value of this threshold depends on the application or the user’s choice. For example, Kovesi’s implementation allows the user to set the error threshold.

For the proposed algorithm implementation, it is slightly more complicated then setting a threshold because the behavior of the proposed algorithm depends on the slope look-up tables. In this proposed implementation, two alternative slope tables are used for arc fitting behavior. The proposed algorithm uses Table 3.2 in modes 0 and 1 and Table 3.4 in mode 2 for line fitting.

According to the Table 3.4, the 4 × 4 pixel line pattern with slope 1 can only be connected to patterns with slope 1. Pattern with slope 2 can only be connected to patterns with slope 2. Pattern with slope 3 can only be connected to patterns with slope 3. Pattern with slope 4 can only be connected to patterns with slope 4. Pattern with slope 5 can only be connected to patterns with slope 5. Pattern with slope 6 can only be connected to patterns with slope 6. Pattern with slope 7 can only be connected to patterns with slope 7. Pattern with slope 8 can only be connected to patterns with slope 8.

Table 3.4. Slope connectivity rules for neighboring patterns (mode-2)

Slope	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	1	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0
3	0	0	1	0	0	0	0	0
4	0	0	0	1	0	0	0	0
5	0	0	0	0	1	0	0	0
6	0	0	0	0	0	1	0	0
7	0	0	0	0	0	0	1	0
8	0	0	0	0	0	0	0	1

The results of the proposed mode-0, mode-1 and mode-2 are shown in Figure 3.25. In mode-0 or the default mode, lines are ignored if they are shorter than two segments like the Helmholtz principle. There is no such restriction in mode-1. In mode-2, Table 3.4 is used for the lowest error threshold and there is very strict fitting of arcs because of no deviation from the 22.5 degree. Mode-2 especially shows that this proposed algorithm does not require validation.

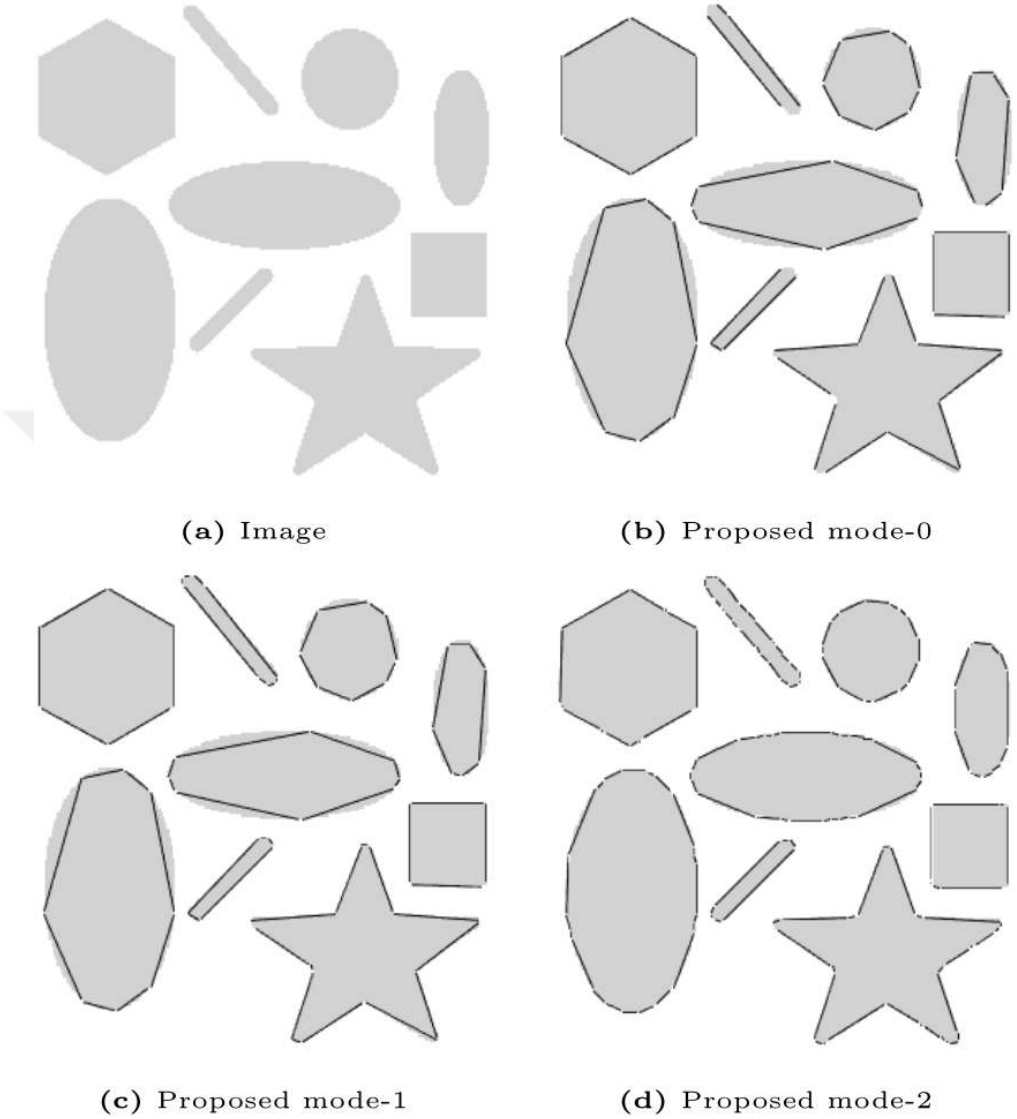


Figure 3.25 (a) A 240 × 240 resolution image with simple geometric shapes. (b–d) Results of the proposed algorithm in modes 0, 1, and 2.

Figure 3.26 shows the result of LSD and EdLines. The proposed algorithm uses a larger threshold while fitting arcs in mode-0 compared with the LSD and the EDLines. There is a criticism (Lu et al., 2015) that LSD and EDLines are dividing arcs into too many small lines seen Figure 3.26.

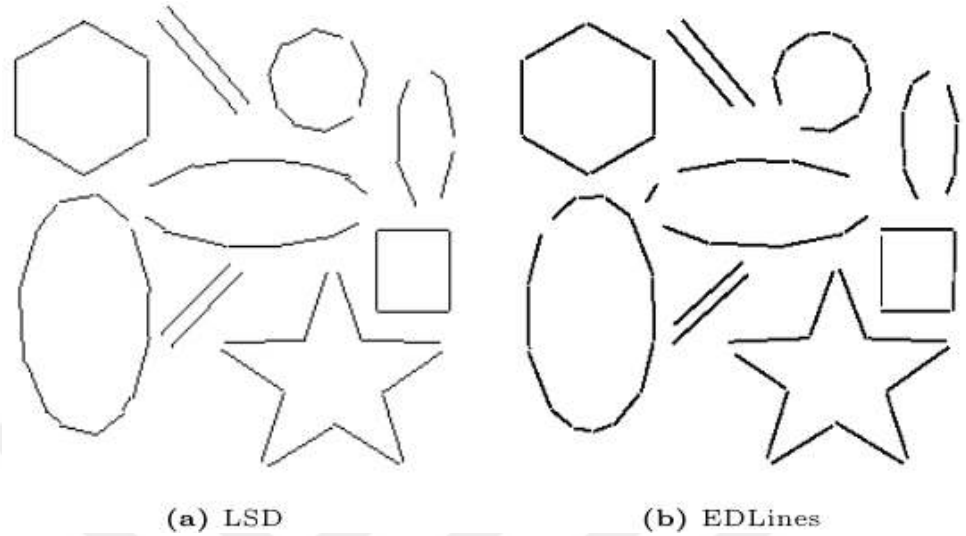


Figure 3.26 Results of (a) LSD and (b) EDLines for the input image in Figure 2.12(a). The pointy tips of ellipses are missing in both of them

In Figure 3.25, the proposed algorithm on top of the original image to verify that the lines are being detected correctly. Some of the lines around the sharp corners of the square are slightly shifted because the low pass filter of the Canny edge detector makes them round. On the other hand, LSD and EDLines have problems with the pointy ellipses and curves.

3.4. Conclusion

The basic aim of this chapter is to explain the segmentation of 110 4 × 4 pixel line patterns.

4. RESULTS AND SPEED TESTS

The proposed algorithm is coded by using MATLAB and C++. All codes are released with step by step explanations and GUI (GitHub, 2022). These explanations consist of the edge detection, piecewise matching (pattern recognition), segmentation of the recognized patterns, showing lines fitted to the segments and verifying the lines with edge pixels. On the GUI, the mode-0 of the proposed algorithm segments minimum two 4 × 4 pixel line pattern without any strict lines, mode-1 is similar to the mode-0 without minimum and finally, mode-2 segments 4 × 4 pixel line patterns with same slope value for the strict line fitting.

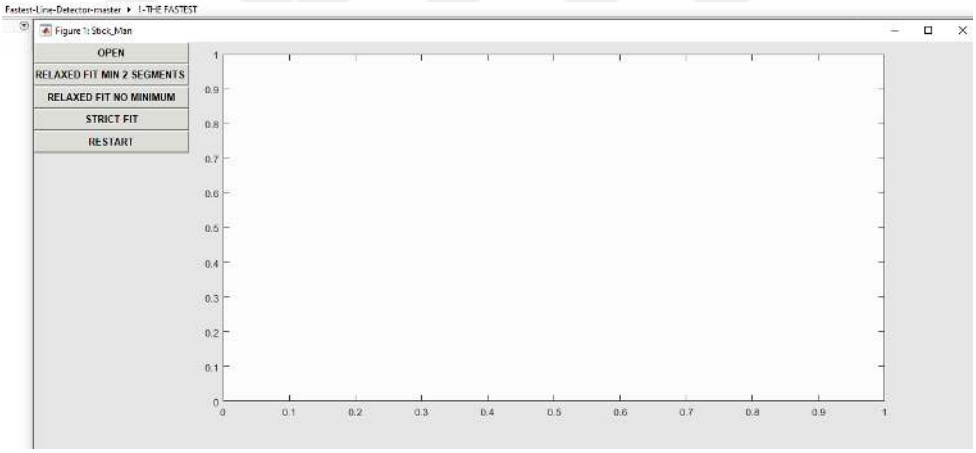


Figure 4.1 GUI of the proposed algorithm for mode-0, mode-1 and mode-2.

The proposed algorithm was initially (Baykal and Yilmaz, 2017) implemented in MATLAB by using the MATLAB executable file (MEX). EDLines was implemented in C++ by using the vector instruction set called the advanced vector extensions (AVX) of the OpenCV library. EDLines has also a Mex file version. Afterward, the proposed algorithm is coded in C++ and compiled to generate a Mex file.

EDLines algorithms has its own edge detector and the edge detection part spends 40% percentage of processing power. This proposed algorithm uses Canny Edge detector but it is not only dependent on Canny, some other faster edge detector could be used instead. In this thesis, the processing time of the edge detector is not included for the speed comparison tests.

Figure 3.25, the proposed algorithm on top of the original image to verify that the lines are being detected correctly. Some of the lines around the sharp corners of the square are slightly shifted because the low pass filter of the Canny edge detector makes them round On the other hand, LSD and EDLines have problems with the pointy ellipses and curves.

Figure 4.2 shows the results of the LSD, the EDLines and the proposed algorithm on the Office_5 image inside MATLAB\toolbox\images\imdata\. According to these results, the proposed algorithm is 17.5 times faster than the EDLines and 160 times faster than the LSD. The speed is tested with 200 iterations elapsed time in seconds for each algorithm.

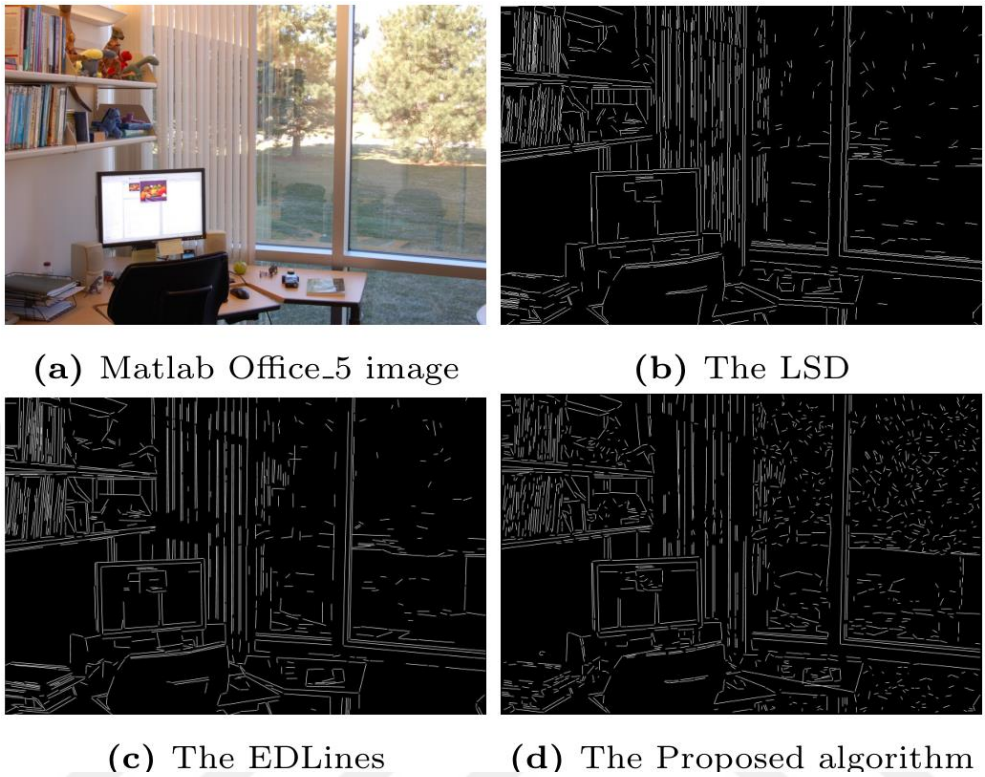


Figure 4.2 (a) The Office_5 image inside MATLAB\toolbox\images\imdata\. (b) Result of the LSD 160 ms.(c) Result of the EDLines 17.5 ms. (d) Result of the proposed algorithm with Canny edge detector ([0.05,0.1],1.2) parameter 1 ms.

Figure 4.3 shows the Jami Mascid image at 1200 × 612 resolution and this image was used to speed comparison for the input of the LSD, the EDLines and the proposed algorithm. Figure 4.4, Figure 4.5 and Figure 4.6 shows the result of LSD, EDLines and the proposed algorithm respectively.

Figure 4.3 The Jami Mascid image at 1200 × 612 resolution.

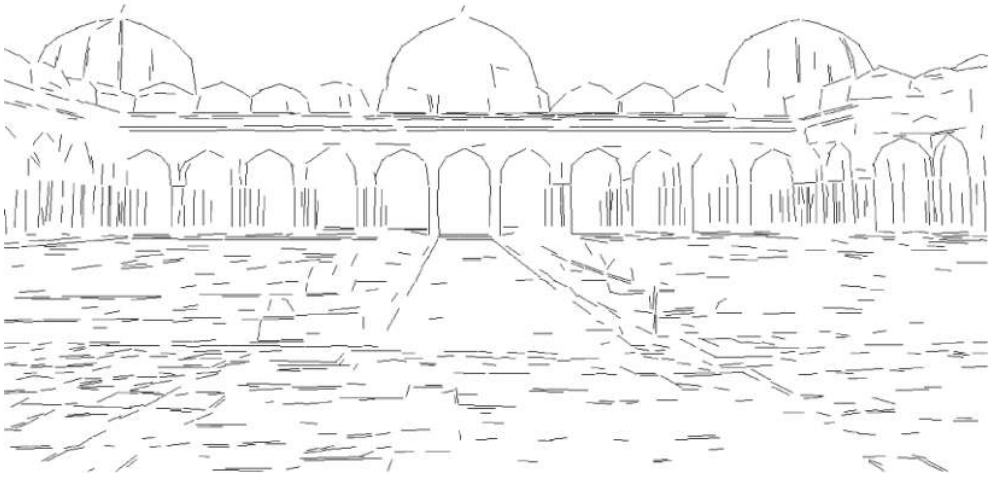


Figure 4.4 The result of the LSD. (145.9 ms).

Figure 4.5 The result of the EDLines. (21.7 ms).

Figure 4.6 The result of the proposed algorithm in mode 0 (1.56 ms).

According to these results, LSD is slower than the EDLines however the EDLines missed some of the small domes because of the EDLines's edge detector. The proposed algorithm is about 14 times faster than the EDLines.

In Figure 4.6 the proposed algorithm uses the Canny edge detector ([0.05,0.1],1.2) for speed comparison. Figure 4.7 and Figure 4.8 show the result of the Canny edge detector ([0.1,0.2],1) and the proposed algorithm for Figure 4.3.

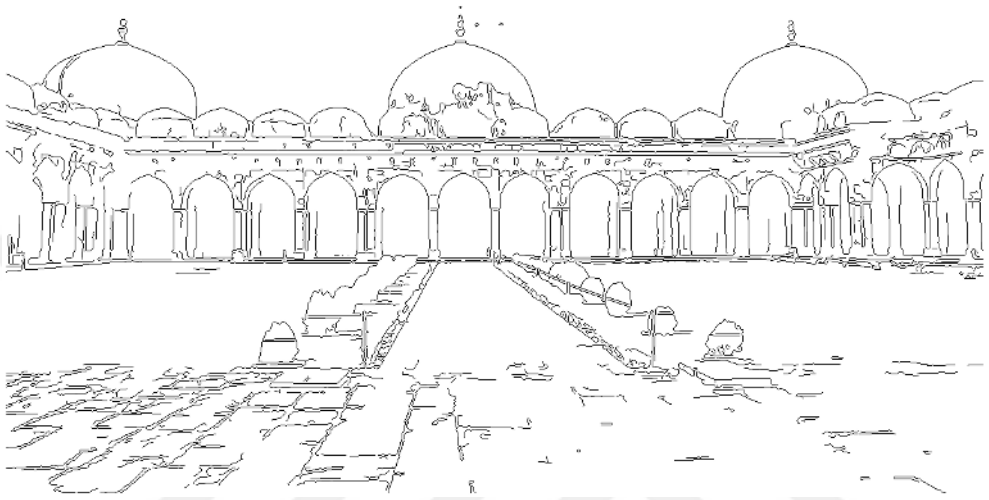


Figure 4.7 The result of the Canny edge detector ([0.1,0.2],1).

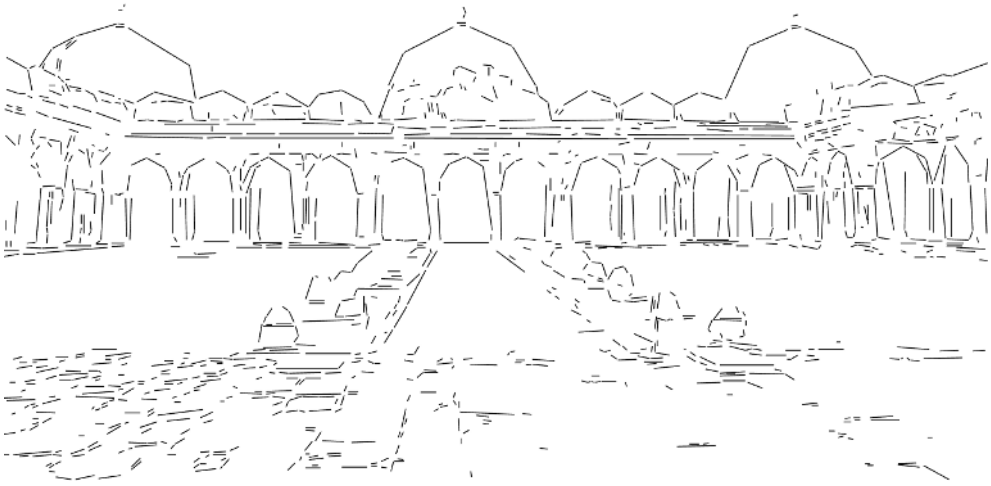


Figure 4.8 The result of the proposed algorithm with Canny edge detector ([0.1,0.2],1).

Figure 4.9, Figure 4.10, Figure 4.11 and Figure 4.12 show four natural images (“Indoor-6”, “Piraeus”, “Pasta” and “Outdoor-9”) and comparisons of the output of the EDLines versus the proposed algorithm. The proposed algorithm also used the edge detector of the EDLines to compare the results versus the Canny edge detector. Figure 4.7 demonstrates that the edge detector of the EDLines tend to pick very settle, almost noise-like details supporting the view of the authors of CannyLines (Lu et al., 2015).

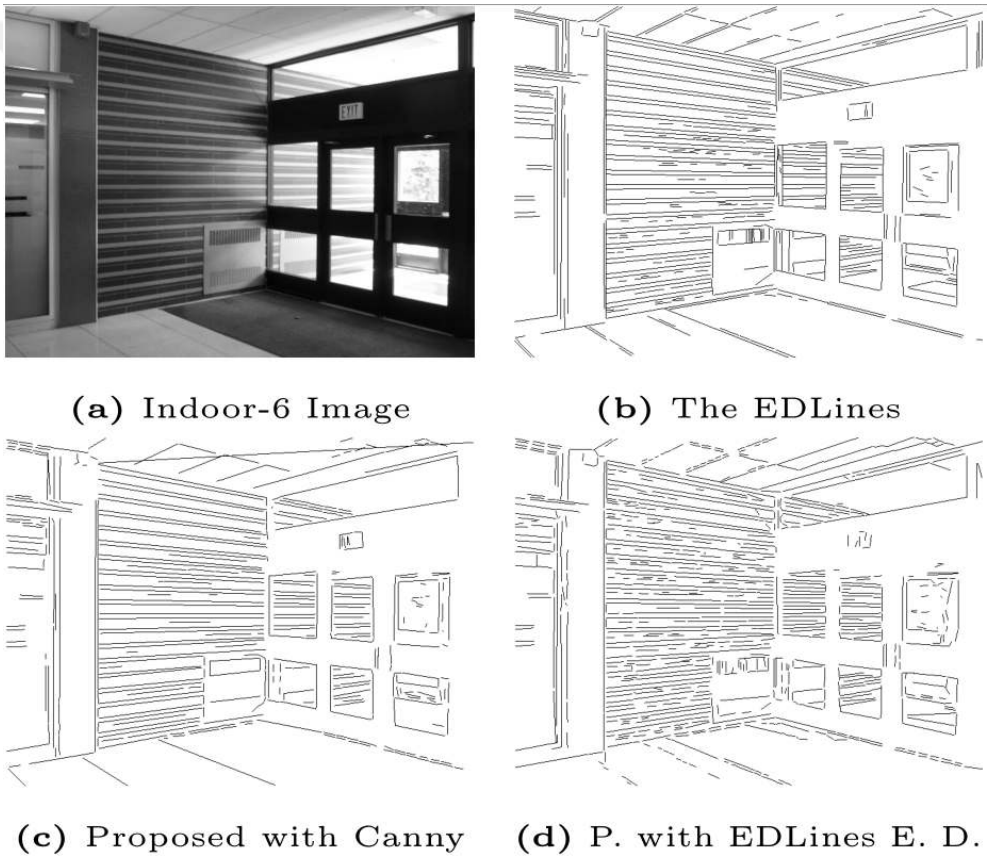


Figure 4.9 (a) Indoor-6 image. (b) Results of the EDLines. (c) Results of the proposed algorithm using the Canny edge detector. (d) Result of the proposed algorithm using the edge detector of the EDLines.

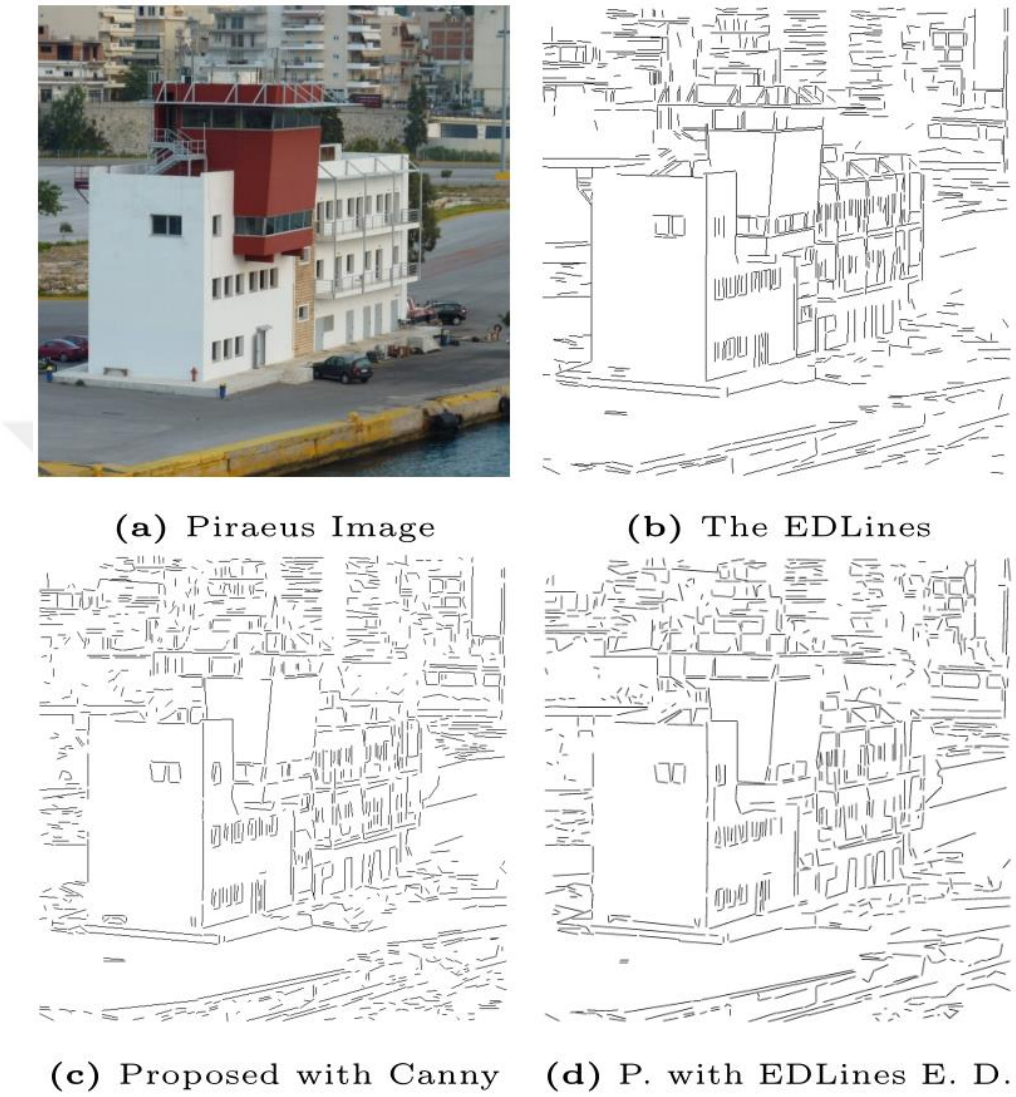


Figure 4.10 (a) Piraeus image. (b) Results of the EDLines. (c) Results of the proposed algorithm using the Canny edge detector. (d) Result of the proposed algorithm using the edge detector of the EDLines.

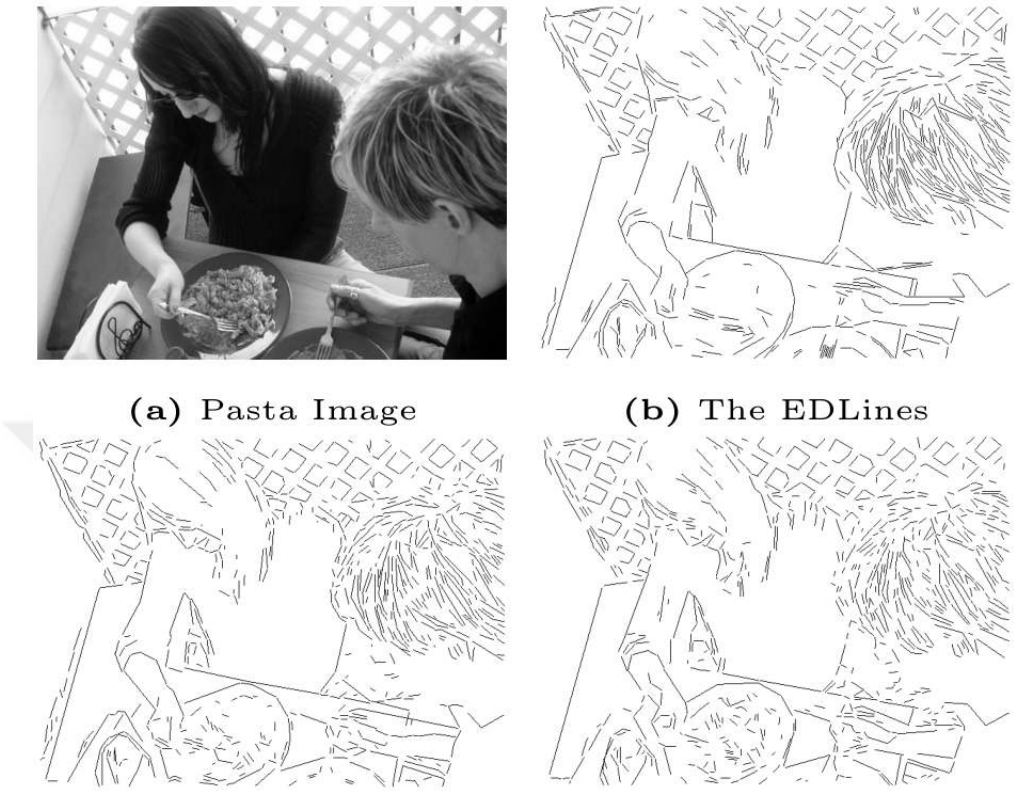


Figure 4.11 (a) Pasta image. (b) Results of the EDLines. (c) Results of the proposed algorithm using the Canny edge detector. (d) Result of the proposed algorithm using the edge detector of the EDLines.

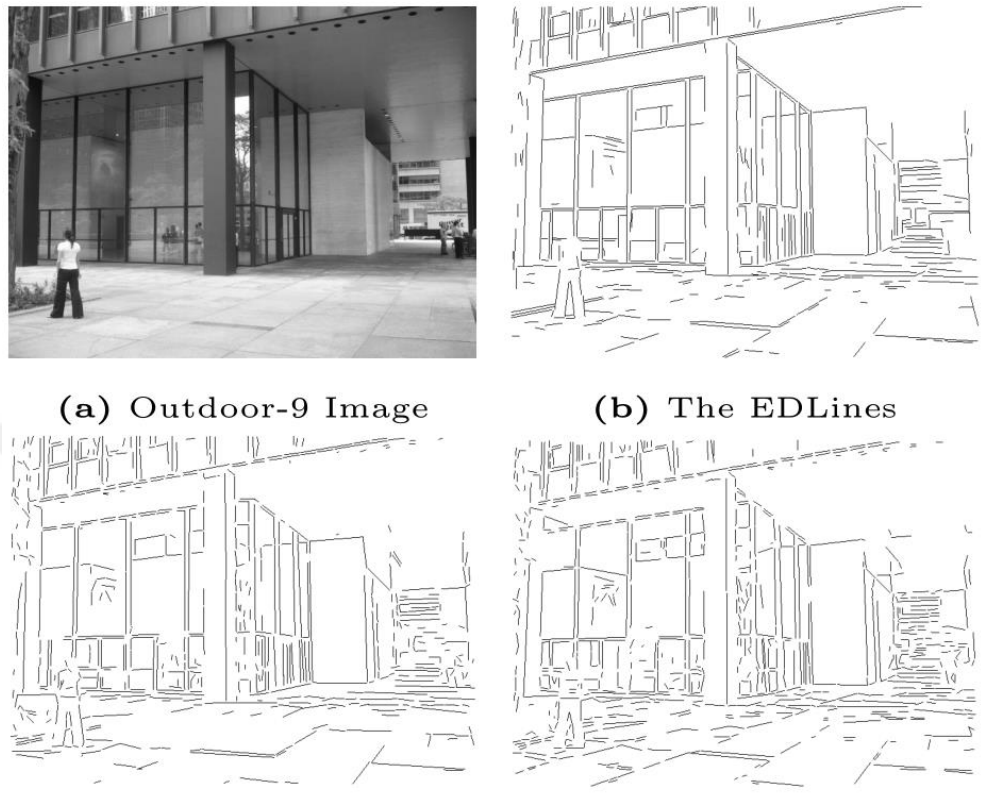


Figure 4.12 (a) Outdoor-9 image. (b) Results of the EDLines. (c) Results of the proposed algorithm using the Canny edge detector. (d) Result of the proposed algorithm using the edge detector of the EDLines.

Table 4.1 shows the results of the speed tests on the images used in this thesis. Figure 4.13 shows the speed results of the LSD, the EDLines, and, the proposed algorithm mode-0 on 10 indoor images and 10 outdoor images from the YorkUrbanDB database (Lu et al., 2022). Total elapsed time for 20 images is 1039.76 ms for LSD, 163.37 ms for EDLines, and 10.98 for the proposed algorithm. According to these results, the proposed algorithm is 14.88 times faster than the EDLines and 94.7 times faster than the LSD. In the literature, there has not been faster line detector algorithm since this proposed algorithm was published (Yilmaz

and Baykal, 2022). A very recent algorithm called LB-LSD (Liu et al., 2019) runs approximately at same speed with EDLines.

Table 4.1 Execution time in ms using C++/MEX files.

Image	Size	LSD	EDLines	Proposed Mode-0
Tree	400 × 400	28.9	7.2	0.36
Boy and girl	480 × 512	53.2	7.54	0.42
House	256 × 256	9.7	1.8	0.07
Office_5	903 × 600	160	17.5	1
JamiMascid	1200 × 612	145.9	21.7	1.56
Indoor-6	640 × 480	44.25	6.45	0.42
Piraeus	600 × 600	70.3	1.34	0.85
Pasta	600 × 450	58.2	8.83	0.57
Outdoor-9	640 × 480	50.75	6.74	0.48

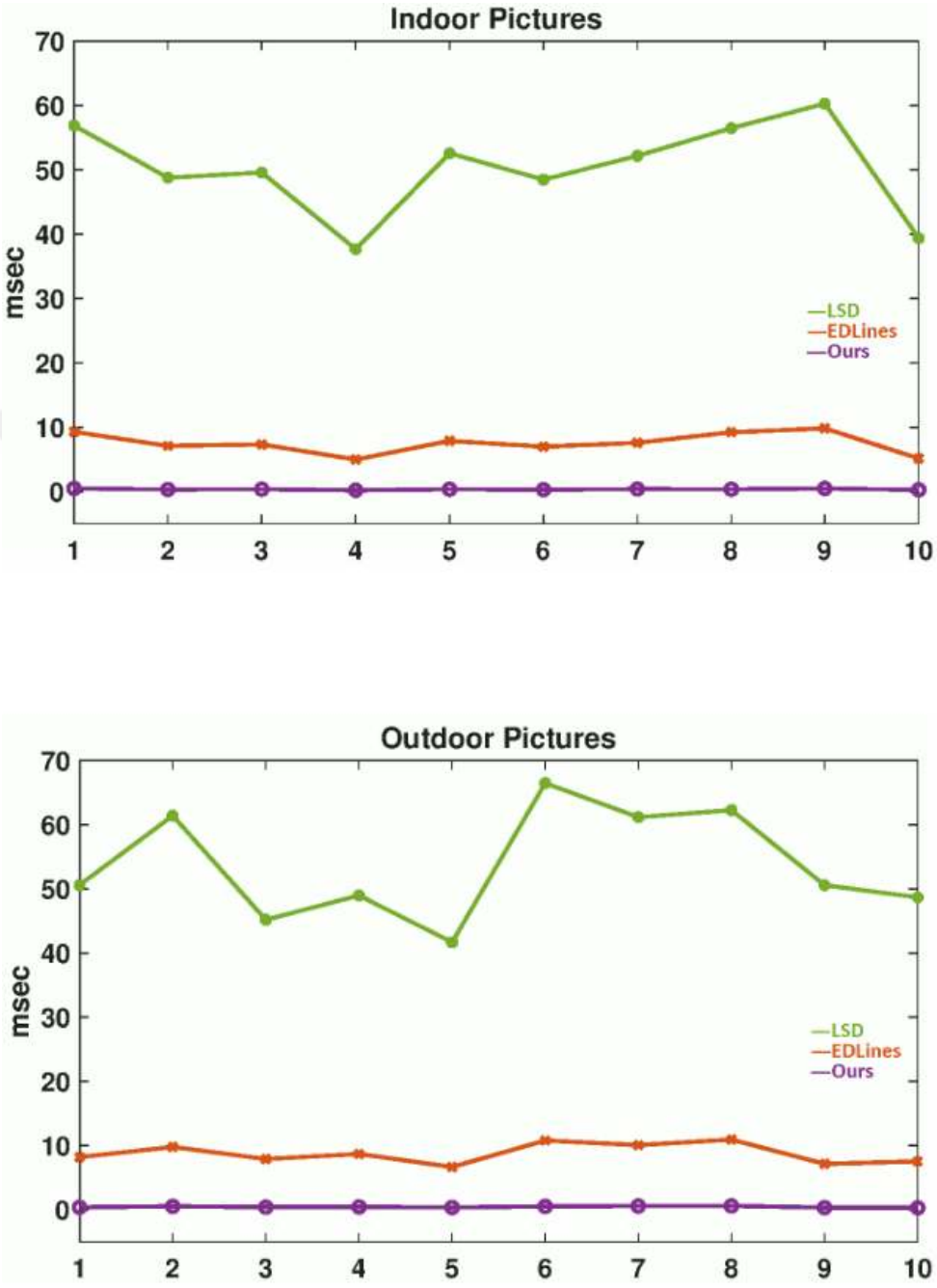


Figure 4.13 10 indoor and 10 outdoor images from the YorkUrbanDB database.

5. CONCLUSION AND FUTURE WORK

5.1. Concluding Remarks and Discussion

This thesis investigates a totally new line detector based on the recognition of 4 × 4 pixel line patterns. Several small look-up tables to decide whether the recognized patterns form a line or not. The maximum size of the look-up table used in this thesis is 64 KB so that the proposed algorithm is so cheap and simple. The test results show that this algorithm is 14.88 times faster than the next fastest line detector, EDLines. In this algorithm, if lines are close to each other <5 pixels, they are not recognized. For this reason, this algorithm is especially designed for high definition image.

5.2. Future Work

After several original contributions have been accomplished in this thesis, various subjects to improve the performance are recommended for future studies as follows:

- Simple and fast edge detectors compatible with the proposed algorithm could be invented.
- Figure 5.1 shows the example of the unrecognized lines close to each other <5 pixels. By updating the look-up table, they will be recognized.
- Some of 4 × 4 pixel line patterns have same slope and tip points. For example; Pattern 5, pattern 7 and pattern 81 in Table 3.1, have same slope value 2, start value 1 and end value 5. So that, look-up table will be optimized.

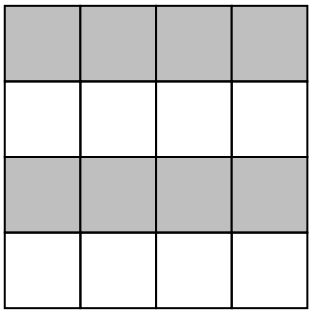


Figure 5.1 Unrecognized lines are close to each other <5 pixels.

REFERENCES

Abdellali, H., Frohlich, R., Vilagos, V., and Kato, Z. (2021). L2D2: Learnable Line Detector and Descriptor. International Conference on 3D Vision (3DV), (pp. 442-452).

Akinlar, C., and Topal, C. (2013). EDLines: A real-time line segment detector with a false detection control. Pattern Recognition Letters, 32(13):1633-1642.

Andrade, D., Bueno, F., Franco, F. R., Silva, R. A., and Neme, J. H., Margraf, E., . . . Amaral, R. D. (2019). A Novel Strategy for Road Lane Detection and Tracking Based on a Vehicle's Forward Monocular Camera. IEEE Transactions on Intelligent Transportation Systems, 20(4):1497-1507.

Baykal, I. C. (2019). Improving the Hough Transform Through Three New Morphological Operators. International Artificial Intelligence and Data Processing Symposium (IDAP) (pp. 1-6). IEEE.

Baykal, I. C., and Yilmaz, I. C. (2017). An extremely fast pattern based line detector. 13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP), (pp. 369-376).

Brown, C. M. (1983). Inherent bias and noise in the Hough transform. IEEE Transactions on Pattern Analysis and Machine Intelligence, 5:493-505.

Burns, J. B., Hanson, A. R., and Riseman, E. M. (1986). Extracting straight lines. IEEE transactions on pattern analysis and machine intelligence, 4;425-455.

Chen, D., Zheng, P., Chen, Z., Lai, R., Luo, W., and Liu, H. (2021). Privacy-Preserving Hough Transform and Line Detection on Encrypted Cloud Images. IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), (pp. 486-493).

Cho, N., Yuille, A., and Lee, S.-W. (2017). A novel linelet-based representation for line segment detection. IEEE transactions on pattern analysis and machine intelligence, 40(5):1195-1208.

Deng, J., Cheng, J., Cai, W., Zhou, Y., Fan, R., and Luo, K. (2021). Property Similarity Line Segment Detector. IEEE International Conference on Imaging Systems and Techniques (IST), (pp. 1-6).

Du, B., Hao, Z., and Wei, X. (2021). Roundness Detection of End Face for Shaft Workpiece based on Canny-Zernike Sub Pixel Edge Detection and Improved Hough Transform. IEEE 11th International Conference on Electronics Information and Emergency Communication (ICEIEC), (pp. 1-4).

Etemadi, A. (1982). Robust segmentation of edge data. 1992 International Conference on Image Processing and its Applications (pp. 311-314). IET.

Fu, Q., Yu, H., Lai, L., Wang, J., Peng, X., Sun, W., and Sun, M. (2019). A robust RGB-D SLAM system with points and lines for low texture indoor environments. IEEE Sensors Journal, 21(19):9908-9920.

GitHub, I. (2022). Fastest-Line-Detector: C++ and Matlab code of the fastest line detector in the literature. Retrieved from https://github.com/icbaykal/Fastest-Line-Detector

Guerreiro, R. F., and Aguiar, P. M. (2012). Connectivity-enforcing Hough transform for the robust extraction of line segments. IEEE Transactions on Image Processing, 21(12):4819-4829.

Hough, P. V. (1962). US Patent No. 3,069,654.

Huang, K., Wang, Y., Zhou, Z., Ding, T., Gao, S., and Ma, Y. (2018). Learning to parse wireframes in images of man-made environments. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, (pp. 626-635).

Huang, S. S., Ma, Z. Y., Mu, T. J., Fu, H., and Hu, S. M. (2020). Lidar-monocular visual odometry using point and line features. IEEE International Conference on Robotics and Automation (ICRA), (pp. 1091-1097).

Illingworth, J., and Kittler, J. (1988). A survey of the Hough transform. Computer vision, graphics, and image processing, 44(1);87-116.

Jiang, X., Ma, J., Xiao, G., Shao, Z., and Guo, X. (2021). A review of multimodal image matching: Methods and applications. Information Fusion, 73:22-71.

Kovesi, P. (2022). Peter's Functions for Computer Vision. Retrieved from http://www.peterkovesi.com/matlabfns/\#edgelink

Li, M., Lafarge, F., and Marlet, R. (2020). Approximating shapes in images with low-complexity polygons. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, (pp. 8633-8641).

Li, X., Cong, Z., and Zhang, Y. (2021). Rail Track Edge Detection Methods Based on Improved Hough Transform. IEEE International Conference on Power Electronics, Computer Applications (ICPECA), (pp. 12-16).

Li, X., Zhang, B., Sander, P. V., and Liao, J. (2019). Blind geometric distortion correction on images through deep learning. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, (pp. 4855-4864).

Lin, G., Tang, Y., Zou, X., Cheng, J., and Xiong, J. (2020). Fruit detection in natural environment using partial shape matching and probabilistic Hough transform. Precision Agriculture, 21:160-177.

Liu, C., Abergel, R., Gousseau, Y., and Tupin, F. (2020). LSDSAR, a Markovian a contrario framework for line segment detection in SAR images. Pattern Recognition, 98:107034.

Liu, H., Li, C., Liu, X., and Wong, T. T. (2022). Neural Recognition of Dashed Curves With Gestalt Law of Continuity. IEEE/CVF Conference on Computer Vision and Pattern Recognition, (pp. 1373-1382).

Liu, Y., Xie, Z., and Liu, H. (2019). LB-LSD: A length-based line segment detector for real-time applications. Pattern Recognition Letters, 128:247-254.

Lu, C., Xia, S., Shao, M., and Fu, Y. (2019). Arc-support line segments revisited: An efficient high-quality ellipse detection. . IEEE Transactions on Image Processing, 29, 768-781., 768-781.

Lu, X., Yao, J., Li, K., and Li, L. (2015). Cannylines: A parameter-free line segment detector. IEEE International Conference on Image Processing (ICIP), (pp. 507-511).

Lu, X., Yao, J., Li, K., and Li, L. (2022). CannyLines: A Parameter-Free Line Segment Detector. Retrieved from {https://cvrs.whu.edu.cn/cannylines/

Luo, S., and Zhao, X. (2022). Application of improved Hough transform in lane line detection. IEEE 10th Joint International Information Technology and Artificial Intelligence Conference (ITAIC). 10:1717-1721.

Matas, J., Galambos, C., and Kittler, J. (2000). Robust detection of lines using the progressive probabilistic hough transform. Computer Vision and Image Understanding, 78(1):119-137.

Meng, C., Li, Z., Bai, X., and Zhou, F. (2020). Arc adjacency matrix-based fast ellipse detection. IEEE Transactions on Image Processing, 29:4406-4420.

Mukhopadhyay, P., and Chaudhuri, B. B. (2015). A survey of Hough Transform. Pattern Recognition, 48(3);993-1010.

OSSIMITZ, C. (2021). Hardware acceleration for line segment detection.. PhD Thesis. Wien.

Ossimitz, C., and TaheriNejad, N. (2021). A Fast Line Segment Detector Using Approximate Computing. IEEE International Symposium on Circuits and Systems (ISCAS), (pp. 1-5).

Tao, C., Mi, L., Li, Y., Qi, J., Xiao, Y., and Zhang, J. (2019). Scene context-driven vehicle detection in high-resolution aerial images. IEEE Transactions on Geoscience and Remote Sensing, 57(10):7339-7351.

Teplyakov, L., Erlygin, L., and Shvets, E. (2022). LSDNet: Trainable Modification of LSD Algorithm for Real-Time Line Segment Detection. IEEE Access, 10:45256-45265.

The MathWorks, I. (2022). Identify peaks in Hough transform - MATLAB houghpeaks. Retrieved from http://www.mathworks.com/help/toolbox/images/ref/houghpeaks.html

Tian, Y., Zhang, C., Jiang, S., Zhang, J., and Duan, W. (2021). Noncontact cable force estimation with unmanned aerial vehicle and computer vision. Computer-Aided Civil and Infrastructure Engineering, 36(1):73-88.

Topal, C., and Akinlar, C. (2012). Edge drawing: a combined real-time edge and segment detector. Journal of Visual Communication and Image Representation, 23(6):862-872.

Vakhitov, A., and Lempitsky, V. (2019). Learnable Line Segment Descriptor for Visual SLAM. IEEE Access, 7:39923-39934.

Van Veen, T. M., and Groen, F. C. (1981). Discretization errors in the Hough transform. Pattern recognition, 14(1-6):137-145.

Von Gioi, R. G., Jakubowicz, J., Morel, J. M., and Randall, G. (2008). LSD: A fast line segment detector with a false detection control. IEEE transactions on pattern analysis and machine intelligence, 32(4):722-732.

Wu, J., Zhang, L., Liu, Y., and Chen, K. (2021). Real-time vanishing point detector integrating under-parameterized ransac and hough transform. Proceedings of the IEEE/CVF International Conference on Computer Vision, (pp. 3732-3741).

Xu, Y., Xu, W., Cheung, D., and Tu, Z. (2021). Line Segment Detection Using Transformers without Edges. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), (pp. 4255-4264).

Xue, N., Bai, S., Wang, F., Xia, G. S., Wu, T., and Zhang, L. (2019). Learning attraction field representation for robust line segment detection. Proceedings

of the IEEE Conference on Computer Vision and Pattern Recognition, (pp. 1595-1603).

Xue, N., Wu, T., Bai, S., Wang, F., Xia, G. S., Zhang, L., and Torr, P. H. (2020). Holistically-attracted wireframe parsing. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, (pp. 2788-2797).

YENIAYDIN, Y. (2019). Sensor fusion of a camera and 2D LIDAR for lane detection and tracking. Master's Thesis. Middle East Technical University.

Yilmaz, I. C., and Baykal, I. C. (2022). Ultrafast line detector. Journal of Electronic Imaging, 31.4: 043019-1 / 043019-15.

Zhang, S., Kang, X., Duan, P., Sun, B., and Li, S. (2021). Polygon structure-guided hyperspectral image classification with single sample for strong geometric characteristics scenes. IEEE Transactions on Geoscience and Remote Sensing., 60(1):1-12.

Zhang, Y., Wang, H., and Yao, R. (2021). A Fast Sea-sky Line Detection Method Based on EDLines. 2nd International Conference on Electronics, Communications and Information Technology (CECIT), (pp. 48-52).

Zhang, Z., Li, Z., Bi, N., Zheng, J., Wang, J., Huang, K., Luo, W., Xu, Y., and Gao, S. (2019). Ppgnet: Learning point-pair graph for line segment detection. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, (pp. 7105-7114).

Zhao, K., Han, Q., Zhang, C. B., Xu, J., and Cheng, M. (2022). Deep Hough Transform for Semantic Line Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 44(9):4793-4806.

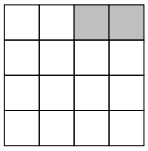
Zhou, Y., Qi, H., and Ma, Y. (2019). End-to-end wireframe parsing. Proceedings of the IEEE/CVF International Conference on Computer Vision, (pp. 962-971).

Zou, D., Wu, Y., Pei, L., Ling, H., and Yu, W. (2019). StructVIO: visual-inertial odometry with structural regularity of man-made environments. IEEE Transactions on Robotics , 35(4): 999-1013.

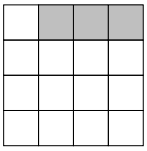
BIOGRAPHY

İsmail Can YILMAZ, ADANA. He received his BSc. from Electrical and Electronics Engineering Department of Middle East Technical University, Ankara, Turkey, in 2008 and MSc from Electrical and Electronics Engineering Department of Cukurova University, Adana, Turkey, in 2011. Now, he has been working as a teacher and research assistant in Electrical and Electronics Engineering Department of Adana A. T. Science and Technology University since 2012. His research interests include computer hardware, FPGA and image processing.

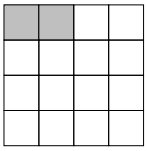
APPENDIX



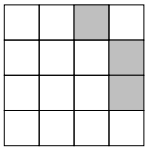
Binary:3
Slope:2
Start:3
End:4



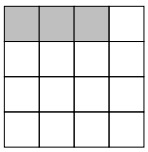
Binary:7
Slope:2
Start:2
End:4



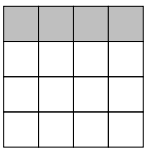
Binary:12
Slope:8
Start:1
End:2



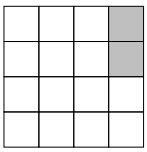
Binary:274
Slope:4
Start:3
End:6



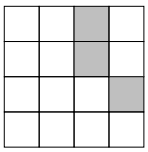
Binary:14
Slope:8
Start:1
End:3



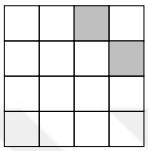
Binary:15
Slope:1
Start:1
End:4



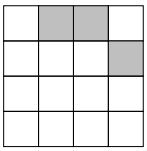
Binary:17
Slope:4
Start:4
End:5



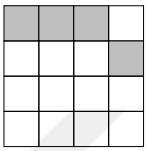
Binary:290
Slope:4
Start:3
End:6



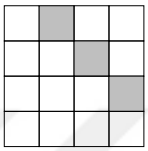
Binary:18
Slope:3
Start:3
End:5



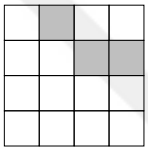
Binary:22
Slope:2
Start:2
End:5



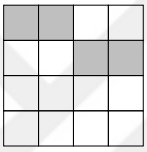
Binary:30
Slope:2
Start:1
End:5



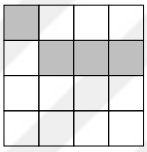
Binary:292
Slope:3
Start:2
End:6



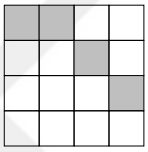
Binary:52
Slope:2
Start:2
End:5



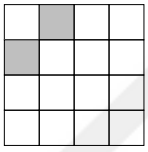
Binary:60
Slope:2
Start:1
End:5



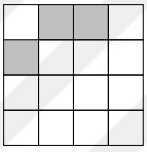
Binary:120
Slope:2
Start:1
End:5



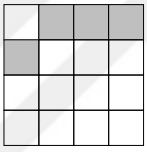
Binary:300
Slope:2
Start:1
End:6



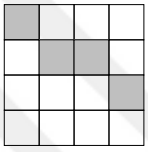
Binary:132
Slope:7
Start:12
End:2



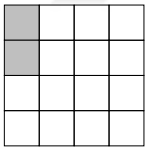
Binary:134
Slope:8
Start:12
End:3



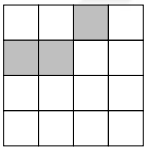
Binary:135
Slope:8
Start:12
End:4



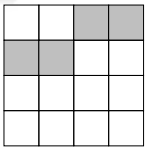
Binary:360
Slope:2
Start:1
End:6



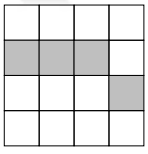
Binary:136
Slope:6
Start:1
End:12



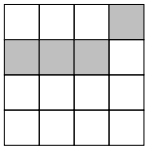
Binary:194
Slope:8
Start:12
End:3



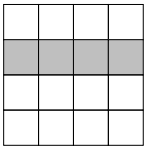
Binary:195
Slope:8
Start:12
End:4



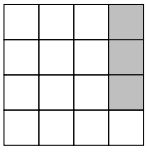
Binary:480
Slope:2
Start:12
End:6



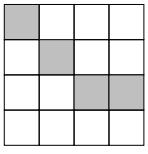
Binary:225
Slope:8
Start:12
End:4



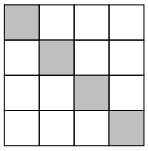
Binary:240
Slope:1
Start:12
End:5



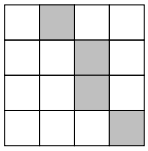
Binary:273
Slope:4
Start:4
End:6



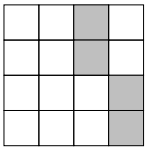
Binary:840
Slope:2
Start:1
End6



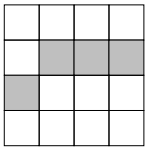
Binary:4680
Slope:3
Start:1
End:7



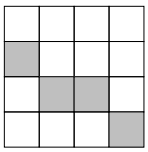
Binary:4644
Slope:4
Start:2
End:7



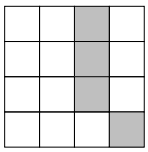
Binary:4386
Slope:4
Start:3
End:7



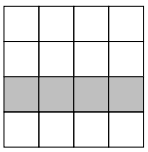
Binary:2160
Slope:8
Start:11
End:5



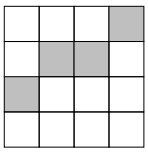
Binary:5760
Slope:2
Start:12
End:7



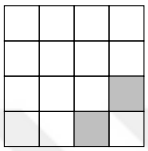
Binary:4642
Slope:4
Start:3
End:7



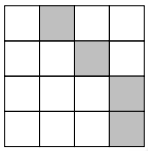
Binary:3840
Slope:1
Start:11
End:6



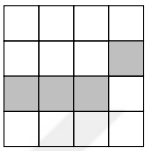
Binary:2145
Slope:8
Start:11
End:4



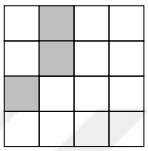
Binary:8448
Slope:7
Start:8
End:6



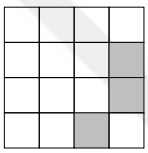
Binary:4388
Slope:4
Start:2
End:7



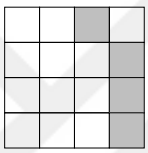
Binary:3600
Slope:8
Start:11
End:5



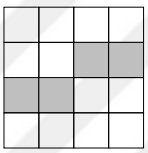
Binary:2116
Slope:6
Start:2
End:11



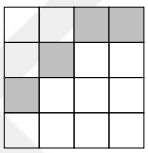
Binary:8464
Slope:6
Start:5
End:8



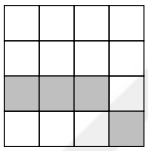
Binary:4370
Slope:4
Start:3
End:7



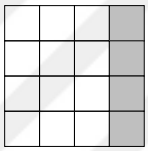
Binary:3120
Slope:8
Start:11
End:5



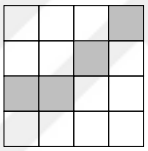
Binary:2115
Slope:8
Start:11
End:4



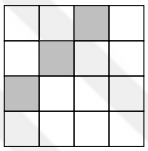
Binary:7680
Slope:2
Start:11
End:7



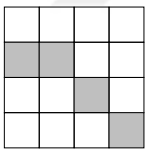
Binary:4369
Slope:5
Start:4
End:7



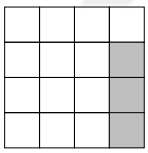
Binary:3105
Slope:8
Start:11
End:4



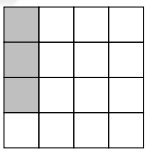
Binary:2114
Slope:7
Start:11
End:3



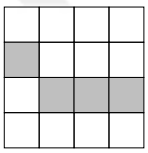
Binary:4800
Slope:2
Start:12
End:7



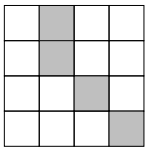
Binary:4368
Slope:6
Start:5
End:7



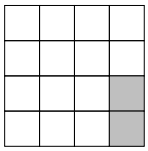
Binary:2184
Slope:6
Start:1
End:11



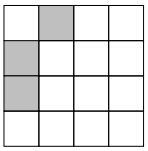
Binary:1920
Slope:2
Start:12
End:6



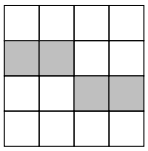
Binary:4676
Slope:4
Start:2
End:7



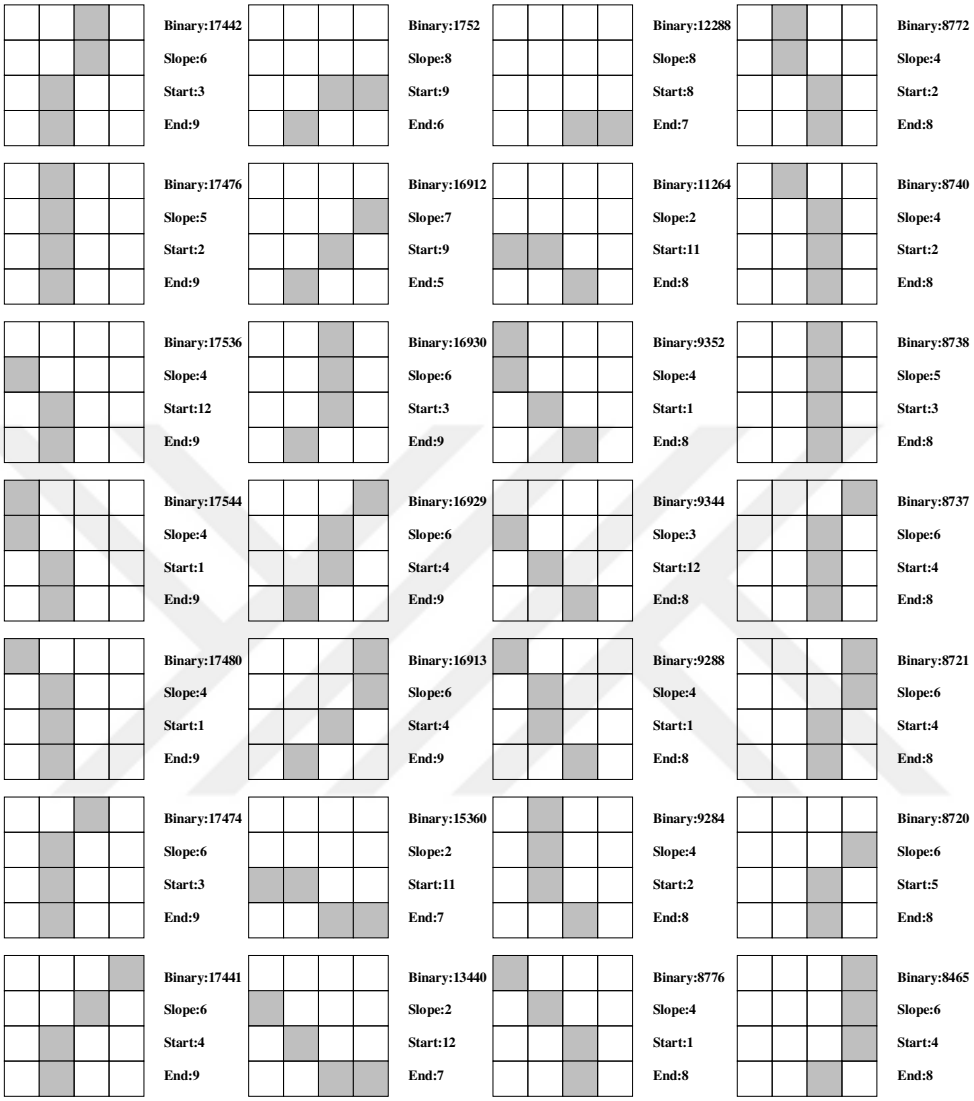
Binary:4352
Slope:6
Start:6
End:7

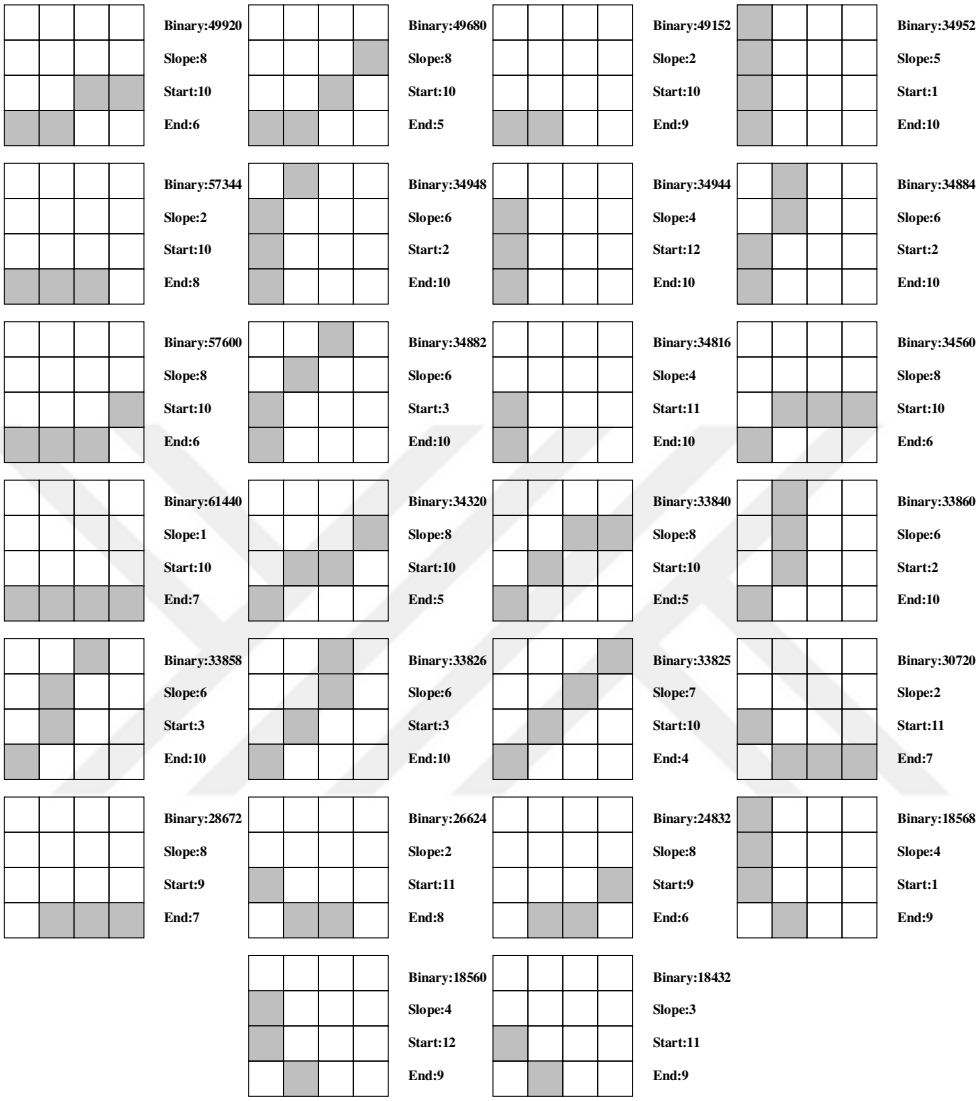


Binary:2180
Slope:6
Start:2
End:11



Binary:960
Slope:2
Start:12
End:6


Binary:17442
Slope:6
Start:3
End:9
Binary:1752
Slope:8
Start:9
End:6
Binary:12288
Slope:8
Start:8
End:7
Binary:8772
Slope:4
Start:2
End:8
Binary:17476
Slope:5
Start:2
End:9
Binary:16912
Slope:7
Start:9
End:5
Binary:11264
Slope:2
Start:11
End:8
Binary:8740
Slope:4
Start:2
End:8
Binary:17536
Slope:4
Start:12
End:9
Binary:16930
Slope:6
Start:3
End:9
Binary:9352
Slope:4
Start:1
End:8
Binary:8738
Slope:5
Start:3
End:8
Binary:17544
Slope:4
Start:1
End:9
Binary:16929
Slope:6
Start:4
End:9
Binary:9344
Slope:3
Start:12
End:8
Binary:8737
Slope:6
Start:4
End:8
Binary:17480
Slope:4
Start:1
End:9
Binary:16913
Slope:6
Start:4
End:9
Binary:9288
Slope:4
Start:1
End:8
Binary:8721
Slope:6
Start:4
End:8
Binary:17474
Slope:6
Start:3
End:9
Binary:15360
Slope:2
Start:11
End:7
Binary:9284
Slope:4
Start:2
End:8
Binary:8720
Slope:6
Start:5
End:8
Binary:17441
Slope:6
Start:4
End:9
Binary:13440
Slope:2
Start:12
End:7
Binary:8776
Slope:4
Start:1
End:8
Binary:8465
Slope:6
Start:4
End:8


Binary:49920
Slope:8
Start:10
End:6
Binary:49680
Slope:8
Start:10
End:5
Binary:49152
Slope:2
Start:10
End:9
Binary:34952
Slope:5
Start:1
End:10
Binary:57344
Slope:2
Start:10
End:8
Binary:34948
Slope:6
Start:2
End:10
Binary:34944
Slope:4
Start:12
End:10
Binary:34884
Slope:6
Start:2
End:10
Binary:57600
Slope:8
Start:10
End:6
Binary:34882
Slope:6
Start:3
End:10
Binary:34816
Slope:4
Start:11
End:10
Binary:34560
Slope:8
Start:10
End:6
Binary:61440
Slope:1
Start:10
End:7
Binary:34320
Slope:8
Start:10
End:5
Binary:33840
Slope:8
Start:10
End:5
Binary:33860
Slope:6
Start:2
End:10
Binary:33858
Slope:6
Start:3
End:10
Binary:33826
Slope:6
Start:3
End:10
Binary:33825
Slope:7
Start:10
End:4
Binary:30720
Slope:2
Start:11
End:7
Binary:28672
Slope:8
Start:9
End:7
Binary:26624
Slope:2
Start:11
End:8
Binary:24832
Slope:8
Start:9
End:6
Binary:18568
Slope:4
Start:1
End:9
Binary:18560
Slope:4
Start:12
End:9
Binary:18432
Slope:3
Start:11
End:9