Department of Computer Science

Submitted in part fulfilment for the degree of BSc

Identifying Game Board and Piece Locations For “Ticket to Ride”

Owen Pantry

30 April 2019

Supervisor: Ian Gray

Word Count: 11,209

Number of Marked Pages: 28

Executive Summary

When playing the physical version of the Ticket to Ride board game scoring can become an issue, as it needs to be kept track of accurately throughout gameplay. Doing this this manually errors occur frequently, causing time to be wasted in recalculating the score. The aim of this project is to produce a method to extract the game board and detect game pieces from an image of a given game state. These two components need to focus on both accuracy and simplicity of use in order to make the games scoring easier to play and score. Image analysis is a hard task to achieve equivalent performance to that of humans, as computers cannot us background knowledge, so different methods will be investigated to ensure the best accuracy is obtained.

The first component investigates 6 different methods, using varying combinations of both colour and shape feature extraction techniques to extract the gameboard from an image before contour finding is applied in order to find the best method. The implemented methods are as follows:

1. A Sobel Operator with an Otzu threshold
2. A Laplacian Operator with a Tozero threshold
3. Image segmentation via colour
4. Canny Edge Detection
5. A combination of methods 1 and 4
6. A combination of methods 3 and 5

For the second component we will look at a method for game piece detection making use of the k-means clustering algorithm. Iterating through each possible game tile region comparing the dominant colour detected via k-means to the expected tile colour in order to determine the presence of game pieces. To ensure the best accuracy two different colour models will be investigated for this implementation – HSV and RGB.

The results of testing the various methods outlined for component 1 show us a combination of image features combined in methods 5 and 6 provide the most accurate and robust approach. With mean errors of 9.78 (95% CI +/-2.57) and 13.87% (95% CI +/-5.1) respectively, along with 97.14% and 100% board detection success based on the test image dataset respectively. Results from method 3, using colour alone to extract image features were shown to be too unreliable for the task with board detection success of 42.85%. Although the other methods, where successful their ability while promising, does not match that of methods 5 and 6. The results from testing our method for component 2 show us that the HSV colour model is optimal with an accuracy of 98% (95% CI +/-2.2) and average number of errors off 0.64 in comparison to RGB which had 87% (95% CI +/-10.1) and 3.64.

In conclusion the approaches portrayed within this paper meet the requirements of both components outlined in terms of ease of use, accuracy and execution time. Through testing we witnessed methods using multiple image features produced better results for finding a board game within an image. We also found that the use of k

means to find the dominant colour within a region of the board produces good results for game piece detection.

Future work that could be undertaken could make use of machine learning with a convolutional neural network as a means to improve robustness of the system. This would need a set of ladled testing and training images in order to be developed, however. An attempt to improve component 2 results further could be made by using an additional image as input in order to produce dynamic colour ranges for use with the end game image, this may however affect ease of use with environmental conditions needing to remain constant (such as light and background).

There are no ethical implications brought about by this project. There was no use of either human participants or personal data within the project. In addition, all images where produced by me unless explicitly stated otherwise. All data displayed is shown without modifications and was produced by me, unless explicitly stated otherwise.

Acknowledgments

I would like to thank Dr. Ian Gray for his guidance and support throughout this project. I would also like to thank the staff members within the University of York Computer Science Department for the many hours of teaching provided over the course of my time at the university, providing me with knowledge used throughout the production of this project.

Table of Contents

1. Introduction 6
   1.1. Approach and Format 6
2. Literature Review 7
   2.1. Image Processing Techniques 7
      2.1.1. 2D Convolution (Blurring) 7
      2.1.2. Morphology 8
      2.1.3. Contour Finding 9
   2.2. Colour 9
      2.2.1. Colour Models 9
      2.2.2. Histogram 10
      2.2.3. K-means Dominant Colour Extraction 10
   2.3. Shape 11
      2.3.1. Thresholding 11
      2.3.2. Edge Detection 11
   2.4. Existing Systems 14
3. Problem Analysis 16
   3.1. Overview of The Problem 16
   3.2. Identifying the Game Board 17
   3.3. Identifying Board Pieces 17
4. Design and Implementation 18
   4.1. Overview 18
   4.2. Development Tools 18
   4.3. Implementation 18
      4.3.1. Board Detection 19
      4.3.2. Game Piece Identification 25
5. Evaluation 28
   5.1. Game Board Detection Evaluation 28
      5.1.1. Evaluation Method and Metrics 28
      5.1.2. Results and Discussion 29
   5.2. Game Piece Detection Evaluation 31
      5.2.1. Evaluation Method and Metrics 31
      5.2.2. Results and Discussion 31
6. Conclusion 33
   6.1. Project Summary 33
   6.2. Statement of Ethics 33

References 34

Appendix 37

Table of Figures

Figure 1 Examples of image blurring techniques 8
Figure 2 (i) Binary Erosion and (ii) Binary Dilation of Figure 1a 9
Figure 3 (i) RGB colour space model (ii) HSV colour space model 10
Figure 4 Colour Histograms of Fig 1a 10
Figure 5 dominant colour histogram of Fig 1a 10
Figure 6 (i) Mean Threshold, (ii) Gaussian Threshold And (iii) Otsu Threshold 11
Figure 7 Sobel mask's 12
Figure 8: Laplacian mask 12
Figure 9 Example of Hysteresis Thresholding 13
Figure 10 Examples of discussed Edge detection methods 14
Figure 11 Ticket to Ride Europe game board 16
Figure 12 System structure overview 18
Figure 13 Example of implemented noise reduction 19
Figure 14 Example of Method 1 output from input image Figure 11 20
Figure 15 Example of Method 2 output from input image Figure 11 21
Figure 16 3D HSV plot 22
Figure 17 Example of Method 3 output from input image Figure 11 22
Figure 18 Example of Method 4 output from input image Figure 11 23
Figure 19 Example of Method 5 output from input image Figure 11 23
Figure 20 Example of Method 6 output from input image Figure 11 24
Figure 21 Example of image after contour finding 25
Figure 22 Perspective warp of Figure 11 25
Figure 23 Image output after applying game piece detection to a game instance 27
Figure 24 Error Due to Similar Background Colour for Method 3 29
Figure 25 Error Due to Additional Points for Method 5 30

Table of Tables

Table 1 Board detection requirements 17
Table 2 Game piece detection requirements 17
Table 3 Edge detection methods 20
Table 4 Component 2 Colour ranges 26
Table 5 Results of Game board Detection Component 29
Table 6 Results of Game piece Detection Component 31

1. Introduction

“Ticket to Ride” is a board game in which players compete to build train routes between locations on a map. It is currently available in both digital and physical mediums. This project relates only to the physical version of the game.

When playing the physical board game scoring can become an issue, as it needs to be kept track of accurately throughout the duration of the game - when doing this manually errors can arise quite frequently. An image analysis system that can, from an image of a given game state, determine which routes each player has constructed and therefore the score they should have would solve this issue.

Such a system would be split into the following set of tasks to achieve this goal:

1. Locate the board game within the input image and warp the perspective to find the birds-eye view of the board.
2. Locate player pieces on the detected board.
3. Calculate scores associated with each player based on game piece locations.

This project will look at how components 1 and 2 of such an image analysis system could be implemented. Firstly, it will investigate different methods to accurately detect the board game within the image using colour and shape descriptors. Secondly it will look at methods for determining the presence of each players train pieces on the detected board.

It is a very difficult task to reproduce the performance capabilities of the human visual system and its ability to characterize pictures based on their content. For a computer system to interpolate an image correctly it must first extract different features from them, instead of using background knowledge [2]. To solve the problems presented within this paper we will therefore investigate some of the most common visual features colour and shape [2], as well as different processing techniques that can be used to enhance performance. From these different techniques a range of implementations will then be attempted in the design section and evaluated to find the best method – that also ensures the best possible ease of use for potential users.

1.1. Approach and Format

The remainder of this project is structured as follows:

Section 2 will look at relevant literature surrounding different techniques and methods pertaining to the project. Starting with image processing techniques used to both improve accuracy and aid in detection, moving on to colour and shape feature descriptors useful for extracting information from images. The section will conclude with a review of similar projects on the topic. **Section 3** looks at an overview of the problem and an analysis of requirements. **Section 4** begins with an overview of the project’s design/implementation and a discussion of the tools used. The section then concludes with an explanation on how each of the two components is implemented. **Section 5** begins with both an explanation and implementation of analysis for game board detection. This is followed by an explanation and implementation of analysis for game piece detection. **Section 6** concludes the project with a summary and discussion as well as an ethical statement.

2. Literature Review

This section provides an overview of the literature regarding computer vision and image processing. To begin we will look at different processing techniques that can be used to optimize image feature extraction. Moving on we will look at how both Colour and Shape descriptors can be used to extract features from images. Finally, existing systems relating to the problems presented will be reviewed.

2.1. Image Processing Techniques

This section describes different techniques used during image processing, image blurring, morphological transformations and contour finding. These techniques are important for ensuring the performance and accuracy of feature extraction techniques outlined in 2.2 and 2.3, minimizing common problems of feature extraction such as image noise and missing information.

2.1.1. 2D Convolution (Blurring)

Image convolution makes use of a user defined kernel to perform tasks such as removing noise and edge detection (see 2.3.2). It works by iteratively placing this kernel across all possible positions in the image, at each position the central pixel is then replaced with the waited sum of kernel weights and their underlying pixels [19, 22]. The desired result of convolution can be specified by adjusting both kernel weights and size.

One result convolution can achieve is image blurring, which is useful for removing image noise. Examples are averaging, Gaussian, median and Bilateral filters [22].

Averaging filters work by replacing each pixel within an image with the average intensity under the mask, achieving good denoising results with the drawback of smoothing edges [22]. With Gaussian blurring the kernel is a gaussian window and is very effective for removing gaussian noise [22, 23]. With Median blurring the central pixel is replaced with the median of all pixels under the kernel, this is very effective for removing salt and pepper noise [22, 23]. The bilateral filter is very effective at removing noise while preserving edge sharpness but is slower than the previously mentioned filters. It achieves this by considering only pixels with intensity's similar to the central pixel for blurring, preserving edges as they'll have a large intensity variation [23]. Examples of each can be seen in Figure 1.

2.1.1.1. **Non-Local Means Blur**

Obtaining good results from edge detection in image processing requires a good quality input image, meaning noise removal is essential. The methods discussed in section 2.1.1 work well for the removal of noise but are not best suited for the preservation of edges. The non-local means (NL-means) noise removal algorithm attempts to solve this problem. With the NL-means algorithm for every pixel, the denoised value is a mean of all pixels weighted by their similarity to the target pixel [20,21]. This results in a noise removal algorithm that improves clarity and preserves detail(edges) in comparison to local methods [21]. An example of NL-means can be seen in Figure 13.

a) Original image b) Averaging filter c) Median filter

d) Gaussian filter e) Bilateral filter

Figure 1 Examples of image blurring techniques

2.1.2. Morphology

Segmentation is the process of partitioning an image into multiple segments, simplifying its representation (i.e. into foreground and background). When processing images with the goal of segmentation, the outcome commonly contains imperfections because of faults in the original image ranging from noise to lighting and texture to name a few. This can be seen when thresholding in Figure 2. To counteract these imperfections, we use morphological transformations on the output binary images, which are simple non-linear operations on the image shape [28].

The process works through image convolution and the use of set theory [22]. Instead of computing a weighted sum, used in blurring, set operations are performed on pixels under the kernel. The two fundamental techniques that form many other morphological techniques are dilation and erosion [22].

Binary Erosion causes white regions within a binary image to diminish by the width of the kernel. An application of this is the removal of white noise and irrelevant detail from an image [22]. Erosion works by setting the central pixel under the kernel to 1 only if all pixels under the kernel are currently set to 1.

Binary Dilation causes white regions within a binary image to be enlarged by the width of the kernel. An application of this is bridging gaps in segmented images [22]. Dilation works by setting all pixels under the kernel to 0 if no pixel under the kernel is currently set to 1.

Figure 2 (i) Binary Erosion and (ii) Binary Dilation of Figure 1a

2.1.3. Contour Finding

Contours are curves joining continuous points along a boundary of an object in an image. They are often obtained following edge detection (see section 2.3.2), this process is useful as a set of coordinates relating to the contour are returned. This is helpful for shape analysis and object finding.

There are many common ways of calculating contours, one example used by the OpenCV library [28] is Satoshi Suzuki's algorithm [26] in order to find contours through border following with topological structure analysis. This method was found to be faster for larger-scale applications such as document scanning [39] - which is suited to the first component of this projects needs of scanning for the game board.

2.2. **Colour**

This section discusses how colour can be used to extract features and process images. Colour is said to be one of the most important features of images in [2] and is computationally not very demanding [3]. We will look at both different ways to model colour and techniques to extract features, which can be used to solve our problem of game board and piece identification.

2.2.1. Colour Models

Images can be represented by an array of different colour models, each expressing a different range of colours known as a colour space when combined with a mapping function.

RGB (Red, Green, Blue) is the most common colour space for image processing [1, 3, 5]. It encapsulates the human visual systems perception of colour closely - with the human eye containing 3 types of cone (receptors) that respond to different wave lengths of light known as blue, red and green cones [1, 4, 5]. The RGB colour space is hard to perceive [1, 5, 4], so others should be considered. [3] found the HSV colour space to produce more successful results for image processing. HSV (Hue, Saturation, Value) can be mapped to/from the RGB colour model and is much more intuitive to visualise [4]. Visualisations of these models can be seen Figure 3 [5].

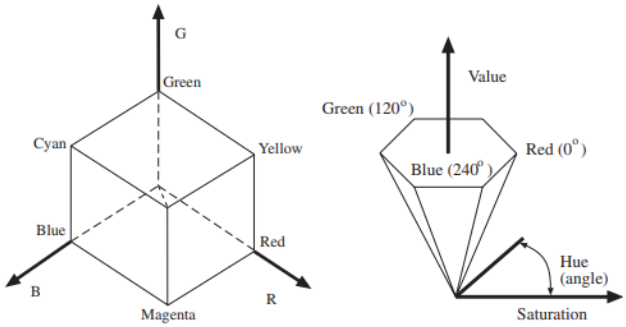


Figure 3 (i) RGB colour space model (ii) HSV colour space model

2.2.2. Histogram

One way in which colour can be used is through histograms [22]. Histograms map the frequency of each colour channels pixel intensity values across the image against the intensity. They have the benefit of being both intuitive to read and simple to compute [2]. However, they're sensitive to noise and contain no spatial characteristics of the image, meaning two semantically different images can have the same histogram [2, 4, 3]. Histograms can be used for both image enhancement and segmentation [22], for example to retrieve the dominant colour within a region. Figure 4 shows different examples of histograms produced from the image in Figure 1a.

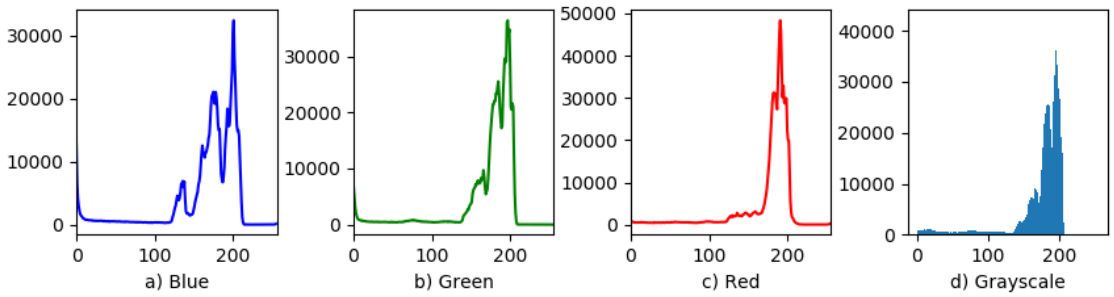


Figure 4 Colour Histograms of Fig 1a

2.2.3. K-means Dominant Colour Extraction

One method to extract the dominant colour from an image or region of interest is through the use of the k-means clustering algorithm. The goal of k-means is to separate a dataset into k cluster/groups, where k is a predefined number [24]. The algorithm works by first calculating k centroids and then maps each point to the cluster with the nearest centroid from it – usually judged from the Euclidean distance. These two steps are then iteratively repeated until the data is optimally split into k clusters.

From k-means we can perform a number of different process from segmenting the image based on these k clusters to determine which colours are dominant [25]. Figure 5 shows an example of the later.

Figure 5 dominant colour histogram of Fig 1a

K means is advantageous in the fact that it is simple to implement. However, its results are heavily dependent on how well the algorithm is initiated – both in initial centroid location and number of clusters k [24, 25].

2.3. **Shape**

Shape, an important cue to distinguish objects for humans [2] is also an important feature that can be used to describe image content [8]. Shape feature extraction can be classified into two groups contour-based methods and region-based methods [2, 8]. Contour based methods look at the boundaries of shapes, whereas Region based methods consider the whole region of the shape [2, 8]. This section looks at different techniques to extract shape features, which can be used to solve our problem of game board and piece identification.

2.3.1. Thresholding

Thresholding, an example of a region-based method, is a simple image segmentation process which is important for low-level vision [10]. In which each pixel in an image is replaced depending on whether its original intensity $I_{i,j}$ is greater or smaller than a given threshold. The value of the replaced pixel is commonly produced through a binary threshold which sets their value to 1 or 0 (black/white), although other methods are available[9-11]. There are two common thresholding types local and global thresholding. A global thresholding technique involves using a single threshold value across the whole image [6, 9-11]. Whereas local thresholding partitions the image into sub images and determines a threshold value for each [6,9-11]. Thresholding simplifies image data making it easier for recognition [9] and is computationally in-expensive and fast [11].

Mean thresholding is a local thresholding technique, in which the threshold T is set to the mean intensity of the local neighbourhood area [9, 11]. Gaussian thresholding is another local thresholding technique, in which the threshold T is the weighted sum of neighbourhood values and the weights are a gaussian window. Otsu thresholding is a global thresholding technique, in which the image is assumed to be bimodal and the threshold T selected minimises the intra-class variance of the two classes [6, 7, 10]. Figure 6 shows an example of each technique discussed.

Figure 6 (i) Mean Threshold, (ii) Gaussian Threshold And (iii) Otsu Threshold of Fig 1a

2.3.2. Edge Detection

Edge detection is an example of a contour-based method to extract information from images. In this section a variety of edge detection methods will be discussed.

2.3.2.1. **Sobel Operator**

The Sobel operator uses gradient to detect edges within an image. It is formed of two 3x3 convolution kernels (Figure 7) which are applied to the image, one responding maximally to horizontal edges and the other vertical, producing two outputs of gradient Gy and Gx.

-1	0	1
-2	0	2
-1	0	1

Gx

1	2	1
0	0	0
-1	-2	-1

Gy

Figure 7 Sobel mask's

The absolute magnitude and edge orientation are then given by [12,16,18]:

$$|G| = \sqrt{Gx^2 + Gy^2} \qquad \theta = \arctan(\frac{Gy}{Gx})$$

The Sobel operator has the benefits of finding both edges and their orientation but is sensitive to noise and often inaccurate [12].

2.3.2.2. **Laplace Operator**

The Laplace operator uses the second order derivative to detect edges at zero crossings [22,23, 12-14]. Unlike with Sobel, Laplace involves one kernel being convoluted across the image. A common kernel that is used is shown in Figure 8.

-1	-1	-1
-1	8	-1
-1	-1	-1

Figure 8: Laplacian mask

The Laplacian L(x, y) of an image with Intensity values I(x, y) is calculated by [22,12]:

$$L(x, y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

It has the advantage of being cheaper to implement than Sobel as only one mask is used to find the edges, however it does not find the edges orientation and is very sensitive to noise as it takes the second derivative [12-14]. To counteract this sensitivity a Laplacian of Gaussian kernel can be calculated [12-14], to first smooth the image and reduce noise.

2.3.2.3. **Canny Edge Detection**

John Canny proposed an algorithm [Cranny's paper] with the intention to optimise available edge detectors based on three criteria - Maximum Signal Noise Ratio, Good localisation of edges and one response per edge. The algorithm involves the implementation of the following steps [12, 15 ,16, 17]:

1. Smoothing of input image with a Gaussian filter, to remove noise.
2. Calculate the gradient magnitude and direction of the smoothed image, commonly obtained via the Sobel operator.
3. Perform non-maximum suppression to determine potential edges that will be considered.
4. Use hysteresis thresholding to decide which edges are truly edges through double thresholding with values T_{max} and T_{min}. Edges above T_{max} are preserved and those bellow T_{min} are removed. Those edges between both T values are preserved when connected to edges above T_{max}, otherwise they are discarded. This process is shown in Figure 9 [28].

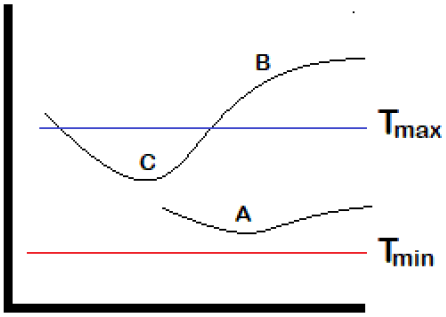


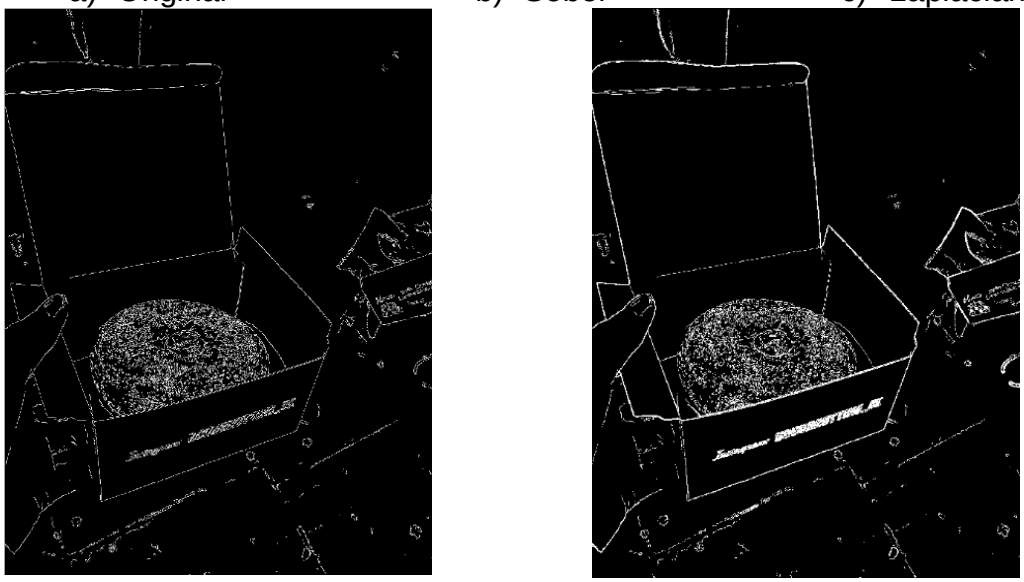
Figure 9 Example of Hysteresis Thresholding A) Discarded B) Preserved C) Preserved

Canny's algorithm has many advantages including better edge detection in noisy conditions, making a good trade-off between noise suppression and localisation [12, 13,15]. However, it has the drawbacks of consisting of complex calculations, being time consuming to compute [12, 13,15] and having manually set threshold values that require prior knowledge to be properly set [15].

Attempts have been made [15,17] to solve the problem of Canny's algorithm requiring manually set thresholds. It suggests that through the use of the Otsu algorithm the upper and lower threshold values used in Canny edge detection can be automatically set. This method uses the variance maximum between two classes as T_{max} and sets T_{min} as 0.5* T_{max}. Experimental results showed an improvement in edge extraction. However, this method assumes an input images histogram has two-peaks - i.e. having both a distinct foreground and background.

An attempt to optimise Canny's algorithm further through a hybrid system has also been researched [18]. Firstly, both Sobel and Canny edge detection are applied to the image separately, before being combined into a hybrid image. Experimental results showed an improvement in accuracy, with the system bringing together advantages from both individual systems.

a) Original b) Sobel c) Laplacian



d) Canny w/Otsu e) Canny + Sobel

Figure 10 Examples of discussed Edge detection methods

2.4. **Existing Systems**

There has been a lot of research into detecting game boards for various games from chess [36-8] to Go [32, 34-5], meaning various techniques have literature pertaining to the topic. This section aims to discuss these varying methods for board game detection present in current research and examine their performance. The research discussed does not represent all methods as the scope of this project focuses on non-learning methods for board detection. This is because non-AI approaches have the benefit of being faster and easier to implement.

[33] Aims to detect the region of the board game within an image through the use of colour markers placed on each corner of the board. Colour segmentation is implemented, making use of the HSI colour space in combination with image thresholding based on the hue component to find corner coordinates. Using this method for board detection provided 93.75% accuracy, with errors arising from board glare. [35, 37] both looked at similar systems to determine the game board location

from a video on a fixed camera. Where board corners are input by the user instead of through the use of colour markers. This approach wouldn't be ideal for this project as we are working with individual images, which can be taken from changing perspectives throughout a game – meaning the user would have to continuously enter corner points.

[34] has a different approach to board detection for a game of Go. Detection focuses on using a Hough transform to detect strong lines in the input image after a Laplacian filter is applied to reduce noise. RANSAC (RANdom Sample and Consensus) is then used to find the board location in Hough space. Using this method for board detection resulted in 100% accuracy for board detection over 55 images. Efforts to reduce computation time where made with detrimental effects to accuracy. This implementation assumed a high contrast between board and background with no game pieces on the board. This method may not as suitable for this project, as go consists of many parallel lines if one is missed for any given reason detection is still feasible. However, if an edge of our board is missed for a given reason the board will not be found.

In [32] the board is found by detecting edges, then lines, and then a grid. First, edges are located with a Laplacian filter. Lines are then found with a Hough transform line detector and split into two groups based on orientation using K-means. RANSAC is then applied to find a sub-grid, this sub-grid is then greedily grown until the full-size board is detected. In all 4 tested games the board was located correctly but there's a lack of testing evident within the report.

[38] looked at both line and corner-based methods for segmentation of a chessboard. The results of the methods discussed are not good enough to recognize chessboards with pieces on them. However, our game would not be affected by this drawback as game pieces do not supersede the edge of game board.

3. Problem Analysis

This section considers the scope of this the project. This involves an analysis and discussion of the problem presented and a discussion of the requirements of the project. In section 3.1 an overview of the problem and different assumptions of the input. In sections 3.2-3 we look at the requirements of the two individual components considered within this project and how they'll be analysed.

3.1. Overview of The Problem

The goal of this project is to investigate how components of a system to score a game of Ticket to Ride can be implemented in order to make the game easier for players to enjoy. More specifically these components are identifying the game board within an input image and detecting the presence of train pieces on the board. The final component was considered outside of the scope of this project as once game pieces are located scoring is an arbitrary calculation in which investigation is not necessary.

The end products focus is on reducing the complications of scoring the game manually, so it needs to be designed with this in mind - we will only consider the input image as input. The use of additional inputs, such as the corner points of the board, would add to the complexity of the system for the user.

To ensure the capturing of the image is uncomplicated the system should not have strict requirements off the perspective and lighting conditions of the image, i.e. the image being birds-eye view. However, some assumptions will have to be made – such as the board pictured is orientated in such a way that up is up, located in the centre of the image as the focus of the image, i.e. fills most of the image. The board perimeter should also not be compromised by another object within the image, but general background noise within the image should not affect results. These restrictions ensure a robust result is achievable without over complicating the process of capturing an image of the game board.

The system should also run in a timely manner. Improving upon the speed of manual scoring, otherwise its usefulness becomes questionable. If scoring is being completed part way through the game, a system that takes too long to run can impede user experience instead of enhancing it.

There are many versions of the Ticket to Ride Game available with different designs and maps. For the purpose of our project we will be considering The Europe game board (Figure 11).

Figure 11 Ticket to Ride Europe game board

3.2. Identifying the Game Board

The task of identifying the game board should, from an input image invariant of lighting and perspective conditions, determine the coordinates of the game board and alter its perspective to ensure it is ready for analysis. From the overview of the problem the following requirements for the task of identifying the game board are:

1	Requirement	Description
a	Identification of coordinates >90% of cases	The component should be able to produce a valid set of coordinates for the game board in 90% or more of input images.
b	Coordinate accuracy of a low mean error and 95% confidence interval of at least +/- 10	The component should be able to produce coordinates for the game board that can be considered accurate and with a 95% confidence interval of less than +/- 10
c	Completion time < 2.5 Seconds	The component should be able to run to completion within 2.5 seconds.

Table 1 Board detection requirements

The values from requirements 1.a-c are arbitrary and determined to be adequate based of end user perception. Ensuring the component performs both in a timely manner and successfully enough to be useful in comparison to manual scoring. They are therefore more of a guideline than hard requirements.

3.3. Identifying Board Pieces

The task of identifying board pieces should look at known game piece locations within an input image and determine the presence of game pieces. The colour of the game piece and the tile it lies on should not affect the system – i.e. the system should be robust against missing game pieces located on tiles of the same colour. From the overview of the problem the following requirements for the task of identifying game pieces are:

2	Requirement	Description
a	Detect game pieces on the game board of differing colours >95% of the time.	The component should be able to correctly detect game pieces on tiles that contain a different colour in >95% of cases.
b	Detect game pieces on the game board of the same colour >95% of the time.	The component should be able to correctly detect game pieces on tiles with the same colour in >95% of cases.
c	Completion time < 10 Seconds	The component should be able to run to completion within 20 seconds.

Table 2 Game piece detection requirements

The values from requirements 2.a-c are again arbitrary and determined to be adequate based of end user perception. Ensuring the component performs both in a timely manner and successfully enough to be useful in comparison to manual scoring. They are therefore more of a guideline than hard requirements.

4. Design and Implementation

In this section the design and implementation of the system is discussed. Section 4.1 begins with a brief discussion of my approach. Section 4.2 follows with my choice of development platform. Finally, sections 4.3 discuss the implementation.

4.1. Overview

The system can be split into three stages board detection, game piece detection and scoring (Figure 12) – here we consider the first two stages.

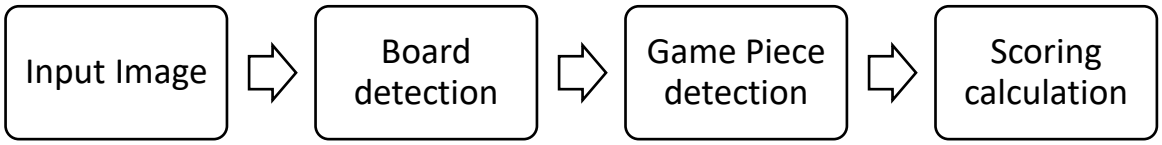


Figure 12 System structure overview

The board detection component looks at differing methods for the detection of the game board based on literature discussed in section 2, with each being evaluated based on their ability to produce output and its accuracy. All methods implemented within this stage go through the same initial pre-processing stage – with the aim of increasing efficiency and robustness of results. The output of this component involves a perspective change to the input image, producing a top down view for the next stage. After board detection the game piece detection component makes use of the top down board to locate player trains.

Splitting the program into two components allows for testing of each component, allowing analysis of changes to method to be carried out.

4.2. Development Tools

The implementation of the proposed system is written in Python 3.6.3 [31]. All code will be completed within Visual Studio Code.

The reason for my decision to use Python is that as well as its standard library there are a number of useful libraries available that fit well with the requirements and aims of the project. For computer vision relayed tasks a wide array of algorithms are supported by OpenCV [28], NumPy [29] is an extremely useful library used for numerical operations and n [30] aids the implementation of machine learning techniques -such as k-means clustering discussed in section 2.

Timing calculations are determined using the timeit [31] library. The modular nature of the design allows both components to be accessed individually.

4.3. Implementation

In the following section the implementation of the system is discussed, split between the two components outlined in section 3.

4.3.1. Board Detection

Although there are no restrictions on either lighting or camera angle, the following assumptions are made on the input image to ensure best possible results:

- The input image is of reasonable quality.
- The board is completely visible within the image.
- The board makes up the majority of the image.
- The board perimeter is not compromised by external objects.
- The board being captured is “Ticket to Ride: Europe” and not an alternative version.

Breaching these assumptions may cause desirable results, but performance will deteriorate drastically.

Pre-processing
Image pre-processing is the first stage, applied identically before each method of board detection and consists of the following tasks. The first task is to reduce the size of the image. The current camera phone standard is 4 megapixels, producing images of 2240x1680 pixels. With larger images leading to drastically increased execution times we can avoid this by reducing images with $x > 1024\ and\ y > 768$ to a width of $x = 1024$, while preserving the ratio of the image. This improves execution time retaining a necessary level of detail – not reducing the image by more than half of the current standard. When testing execution time for method 4, this pre-processing step improved execution time from 31.4 seconds to 2.49 seconds.

As discussed in Section 2.3 edge detection techniques performance suffer greatly from noise within the image, to ensure better success noise reduction is applied. Methods discussed in section 2.1 such as Gaussian blur, Median Blur, Bilateral filtering and fast Non-Local means blurring where all tested. The later method survived the final implementation. The other methods failed due partly to their results varying drastically depending on lighting and background conditions. Another reason was that their methods caused edges to be lost more frequently than NL-means, effecting shape feature extraction. A window size of 11 and a filter strength of 15 proved to be sufficient to remove most noise while retaining edge detail.

Figure 13 Example of implemented noise reduction

Edge Detection

After pre -processing a number of differing methods are applied, as outlined in section 4.3.1.1-5, this is due to a key outcome of the project being to investigate which method works best. An overview of these methods can be seen in Table 3.

Method	Description
1	This method makes use of the Sobel operator to locate edges.
2	This method makes use of the Laplace operator to locate edges.
3	This method makes use of colour features to locate edges
4	This method makes use of the Canny operator to locate edges.
5	This method attempts to optimise methods 1 and 4 through a hybrid approach of the 2.
6	This method attempts to optimise methods 1, 3 and 4 through a hybrid approach of the 3

Table 3 Edge detection methods

4.3.1.1. **Method 1: Using Sobel**

From the pre-processed image a Sobel mask is convolved horizontally and vertically approximating the gradient magnitude by combining both results as discussed in section 2.3.1.1. Through testing different kernel sizes, a 3x3 kernel produced the best results.

The next stage involves thresholding the image in order to produce a binary image ready for contour finding. Both Otsu and Tozero thresholding where tested resulting in the Otsu threshold being used. This was due to a preliminary mean absolute area smaller than Tozero following further steps, as well as a lower rate of failed detection. Otsu thresholding as the advantage here of using an adaptive threshold, meaning it seeks to find the optimal threshold without user interaction.

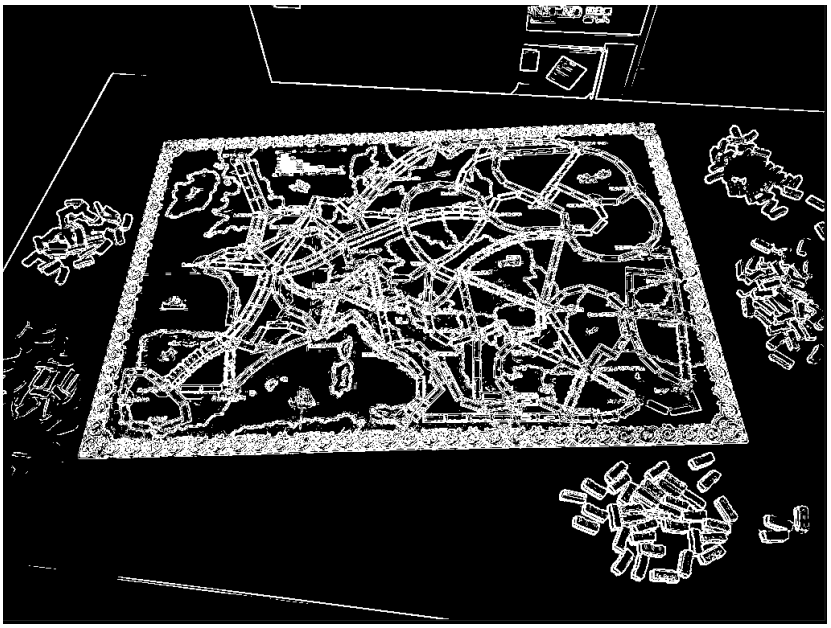


Figure 14 Example of Method 1 output from input image Figure 11

4.3.1.2. **Method 2: Using Laplace**

From the pre-processed image a Laplace operator is convolved across the image calculating the second order derivative as described in section 2.3.1.2. Through testing different kernel sizes, a 9x9 kernel produced the best results.

The next stage again involves thresholding the image in order to produce a binary image ready for contour finding. Both Otsu and Tozero thresholding where tested resulting in the Tozero threshold being used. This was due to a preliminary mean absolute area smaller than Otsu following further steps and an equal rate of failed detection. The threshold value in which optimal results where produced was 30.

Following on from thresholding morphological closing was carried out to fix consistent errors with contour finding in further steps due to the edges within the image not being fully connected. Closing is a compound morphological operation of dilation followed by erosion, both discussed in section 2.1.2, to extend the boundaries of foreground segments. In this instance a kernel size of 3x3 was optimal

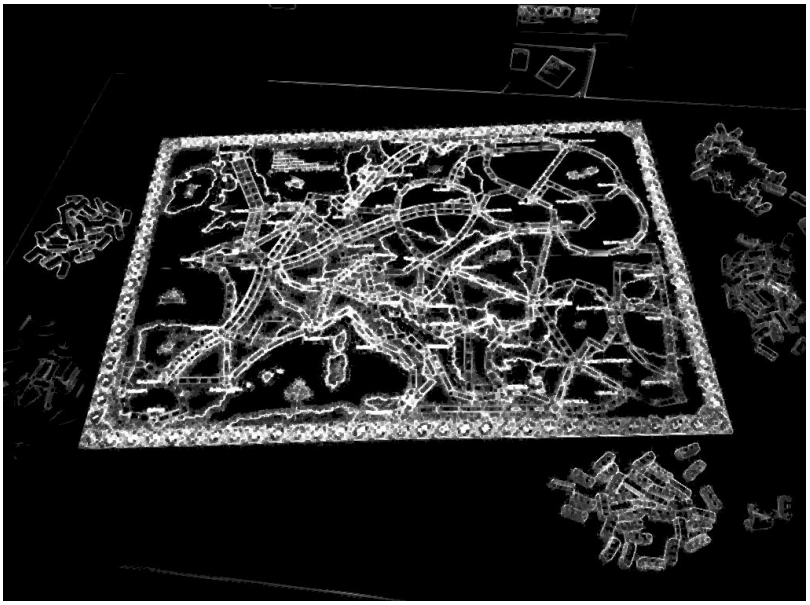


Figure 15 Example of Method 2 output from input image Figure 11

4.3.1.3. **Method 3: Using Colour**

This method makes use of the assumption that the user is playing the “Ticket to Ride: Europe” board, which is predominantly blue in colour (see from Fig 13). To accurately visualise the correct colour range to segment the image via colour we convert the input image from RGB to HSV and create a 3D plot of Hue, Saturation and Variance (Figure 16). From this plot we can see the board is a localised range of blues that can be extracted to produce a lower/upper range to threshold the input image. Through testing the optimal range was found to be a lower range of [85, 1, 1] and an upper range of [135, 190, 200].

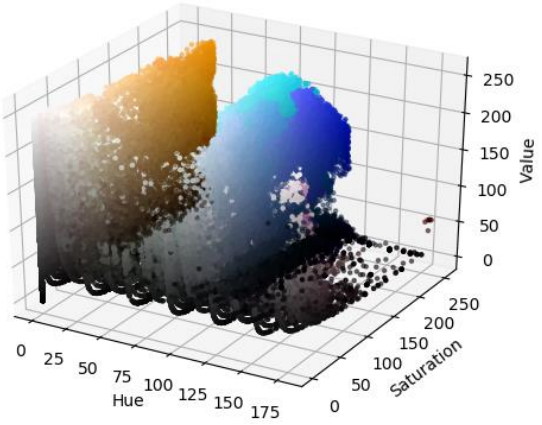


Figure 16 3D HSV plot

Following on from colour thresholding morphological closing was carried out to again fix consistent errors with contour finding in further steps due to the edges within the image not being fully connected. In this instance a kernel size of 13x13 was found to be optimal with large disconnected between edges.

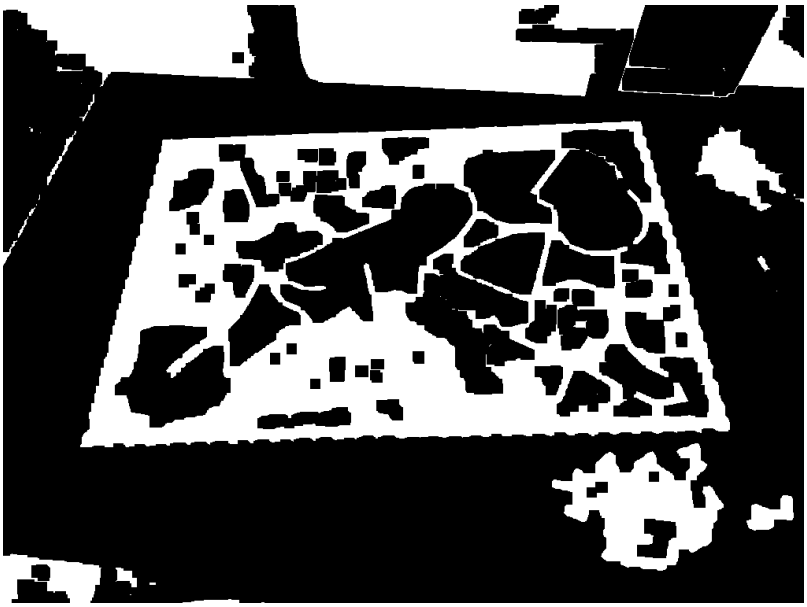


Figure 17 Example of Method 3 output from input image Figure 11

4.3.1.4. **Method 4: Using Canny**

From the pre-processed image Canny edge detection is applied to the image as described in section 2.3.1.3. Different thresholds where tested for T_{max} and T_{min} in order to determine their optimal values – ranging from manual values and automatic values from median and Otsu as proposed by [15, 17]. Using Otsu's method to set thresholds proved produce optimal results out of the three.

Following on from thresholding morphological closing was carried out to fix consistent errors with contour finding in further steps due to the edges within the image not being fully connected. In this instance a kernel size of 3x3 was optimal.

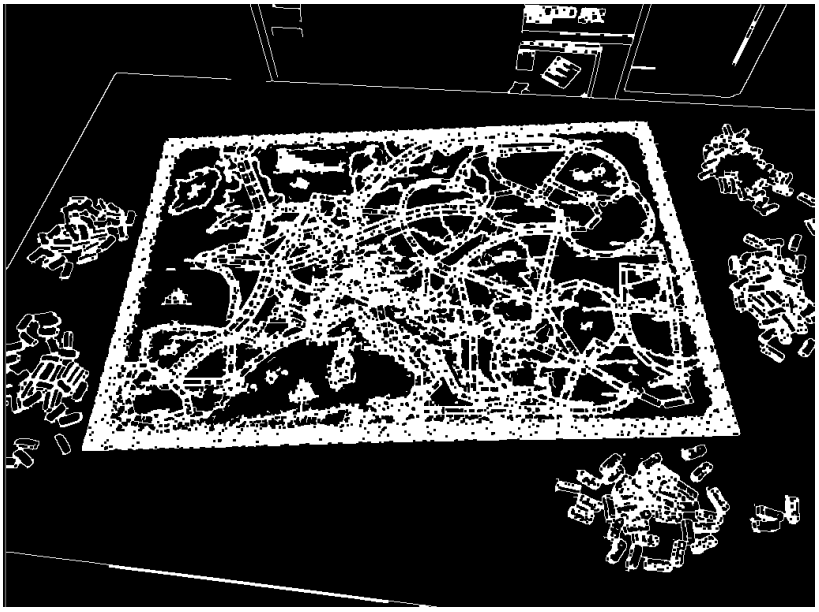


Figure 18 Example of Method 4 output from input image Figure 11

4.3.1.5. **Method 5: Using a combination of Sobel and Canny**

Following on from [18], this method attempts to optimise method 4 through the use of a Sobel/Canny hybrid system. Firstly, both Sobel and Canny edge detection are applied to the pre-processed image separately as define din method 1 and 4 respectively.

The two output images are then combined using a bitwise ‘AND’ to form one image. Bitwise ‘AND’ works by comparing the binary pixel values of both outputs of 1 and 4, if corresponding pixels are both 1 the pixel is set to 1, otherwise it is set to 0. Morphological closing then followed to fix consistent errors with contour finding in further steps due to the edges within the image not being fully connected. In this instance a kernel size of 3x3 was optimal.

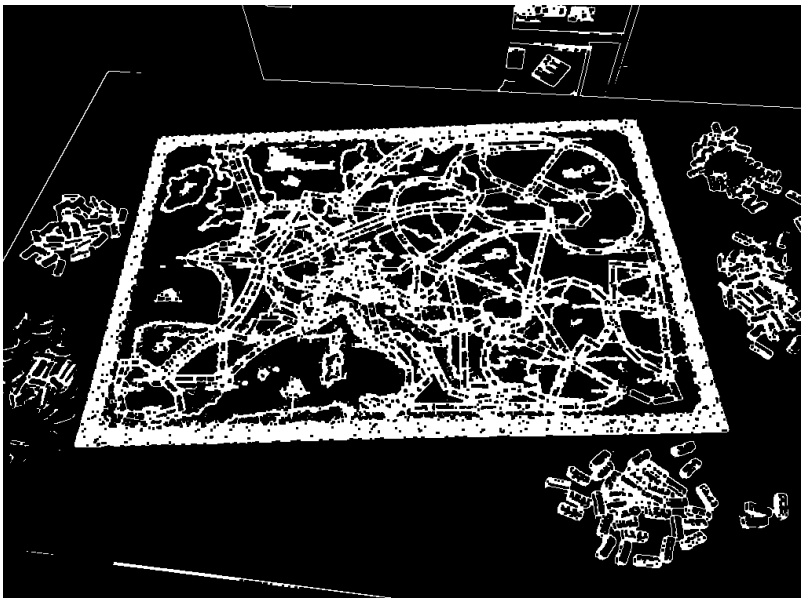


Figure 19 Example of Method 5 output from input image Figure 11

4.3.1.1. **Method 6: Using a Combination of Sobel, Colour and Canny**

Method 6 attempts to produce further optimisation to method 5 through the addition of a colour extraction component in an attempt to improve robustness. The two images output immediately after method 3 and 5 are combined using a bitwise 'OR' to form one image. Bitwise 'OR' works by comparing the binary pixel values of both outputs of 3 and 5, if either of the corresponding pixels are 1 the pixel is set to 1, otherwise it is set to 0. Morphological closing then followed to fix consistent errors with contour finding in further steps due to the edges within the image not being fully connected. In this instance a kernel size of 3x3 was optimal.

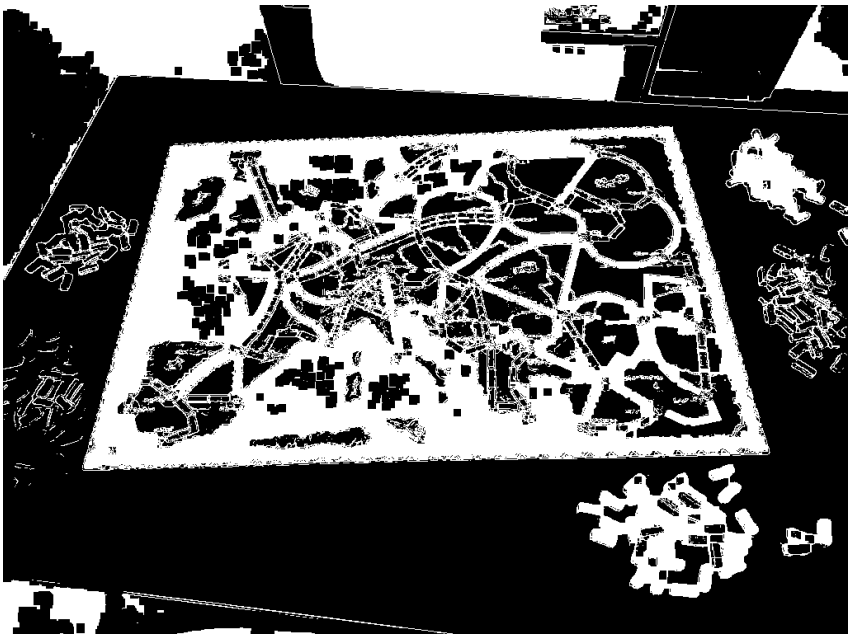


Figure 20 Example of Method 6 output from input image Figure 11

Post Edge Detection

After edge detection is complete, we locate contours within the binary image. For this we make use of the OpenCV library [28] and its built in findContours() operator based on the Suzuki algorithm [26]. We then make use of the assumption that the board fills most of the image by sorting found contours from greatest to smallest, keeping only the largest contours based on area. From this contour we make use of the Ramer-Douglas-Peuker algorithm [27] which from a curve composed of line segments finds a similar curve with fewer points based on a defined precision. It works by recursively dividing the line discarding all points within the defined precision and keep all points greater. The optimal precision found throughout testing was 0.025. The resulting contour is then checked to determine if it is indeed a 4-sided polygon – if it is the image is passed to the next stage, if it is not then board detection can be considered as a failure for the given input image.

Figure 21 Example of image after contour finding

The final stage applied to each of the above methods is a perspective warp, to produce a top down view of the board. This is done in preparation for the next components task of locating game pieces. An example of this process can be seen in Figure 22.

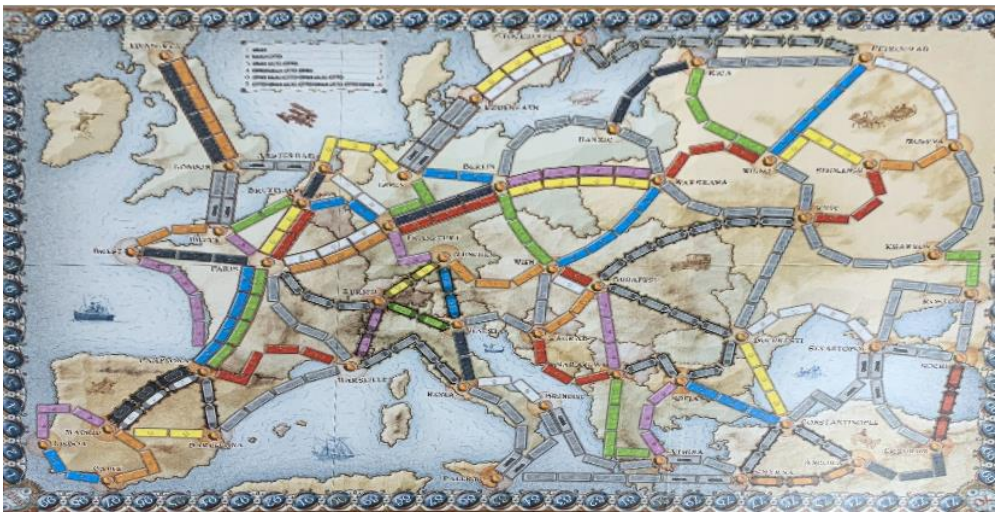


Figure 22 Perspective warp of Figure 11

4.3.2. Game Piece Identification

The input for game piece identification is received from the first component of the system and therefore carries the same assumptions of image quality and board type, “Ticket to Ride: Europe”. The image received from the first component is also assumed to be a perfectly warped birds-eye view image of the game board.

Our assumption that “Ticket to Ride: Europe” is being played provides us with some key information. Firstly, that game piece locations are standardised within each image, meaning that possible game piece locations can be hardcoded as regions of interest (ROI) into the system. Secondly the expected colour of each game tile is known therefore known.

We start by iterating through each ROI on the warped image top down view of the game board. At each iteration we make use of the k-means algorithm, as described in 2.2.3, to retrieve the dominant colour within the ROI. As the expected tile colour is known we can compare the dominant colour calculated to a colour range of the expected tile colour. Tile colour ranges were calculated from an average of all similarly coloured tiles across a number of empty game boards, with the addition of a small margin of area. The colour ranges calculated for each possible tile colour are found in Table 4.

Tile Colour	RGB		HSV	
	Lower	Upper	Lower	Upper
Red	140,80,65	190,115,100	0,30,56	35,58,75
Yellow	170,150,85	202,185,110	38,35,65	60,55, 78.5
Green	130,145,80	155,165,105	63,30,50	93,45,75
Blue	60,112,125	120,155,185	190,20,55	208,70,80
Grey	115,110,100	160,160,155	0,0,40	360,25,73
White	130,130,120	185,170,160	0,0,53	37,24,74

Table 4 Component 2 Colour ranges

In the case where the dominant colour **is not** within the ROIs colour range, we can determine a train is present and record this. In the case where the dominant colour **is** within the ROIs colour range, two possible events occur. Firstly, if the tile is not expected to be one of the possible game piece colours (red, green, yellow, blue), we can determine a train is not present and record this. However, if the tile is expected to be one of the possible game pieces colours, we compare the dominant colour to a stricter colour range to determine if a train is present or not. This works as although the game pieces are the same colour, they are a noticeably different in tone and texture.

During implementation two different colour ranges (RGB and HSV) where implemented in order to compare their effectiveness, both of which are discussed in 2.2.1. To produce HSV values from RGB values obtained from k-means the following formulas are used:

$$Standardisation:\ R' = \frac{R}{255}, G' = \frac{G}{255}, B' = \frac{B}{255}$$

$$C_{max} = \max(R', G', B'), \qquad C_{min} = \min(R', G', B'), \qquad \Delta = C_{max} - C_{min}$$

$$Hue:\ H = \begin{cases} 60 \times \left(\frac{G' - B'}{\Delta} mod 6\right), C_{max} = R' \\ 60 \times \left(\frac{B' - R'}{\Delta} + 2\right), C_{max} = G' \\ 60 \times \left(\frac{R' - G'}{\Delta} + 4\right), C_{max} = B' \end{cases}$$

$$Saturation\colon\ S = \begin{cases} 0\,, & C_{max} = 0 \\ \dfrac{\Delta}{C_{max}}, & C_{max} \neq 0 \end{cases}$$

$$Value\colon\ V = C_{max}$$

An example of the system output where tiles deemed to contain trains have been highlighted can be found in Figure 23.

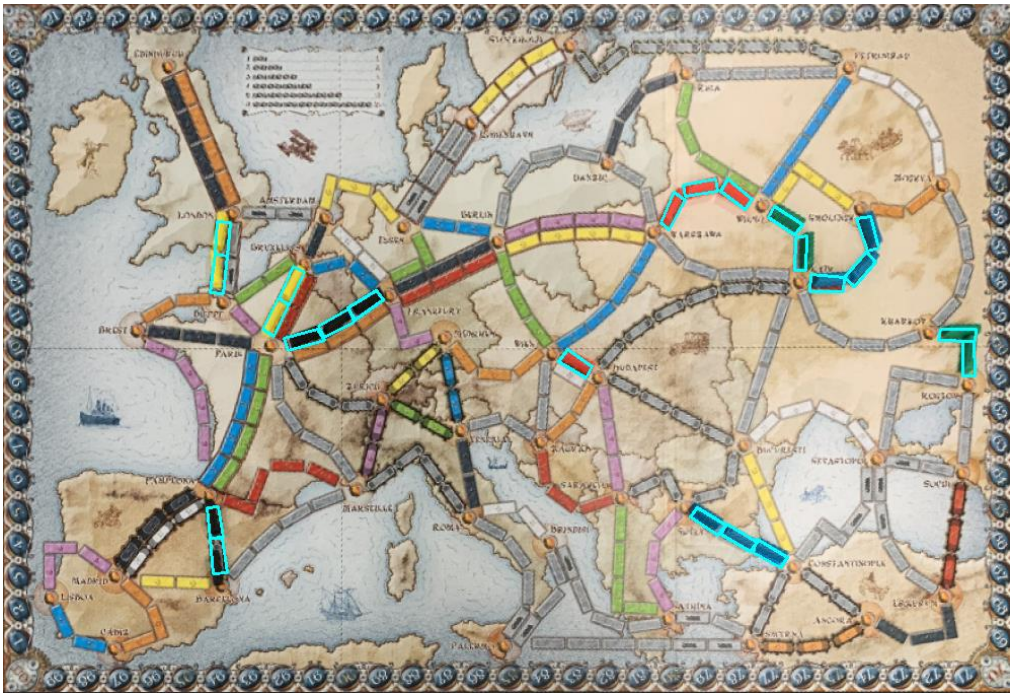


Figure 23 Image output after applying game piece detection to a game instance

5. Evaluation

This section evaluates the entire project. The evaluation Is split into two sections (5.2 and 5.3), in which both system components are evaluated individually. In both sections an outline of the metrics used for performance evaluation of each section is explained, measured and an evaluation of the requirements set out at the start of the project explored.

5.1. Game Board Detection Evaluation

5.1.1. Evaluation Method and Metrics

This section discusses the process behind **game board detection** testing, as well as the metrics used to compare each method. For each method implemented an identical set of 35 images was used to run tests on. These images varied in a number of ways – gameboard background, perspective and lighting (on or off).

To begin testing we manually find the true corner points for each test image, these will be used calculate the accuracy of the coordinates produced by each method. Then I proceeded to run each method on the set of images. if the game board was found both the output board coordinates produced and execution time for each was recorded, if not the image was discarded from the results for the given method.

We now have two sets of coordinates we can use for each method, the true board coordinates and the detected board coordinates. From these values we can calculate the absolute error of real vs detected points for each corner point in an image with equation 1, where (x_t, y_t) are the true coordinates and (x_d, y_d) are the detected coordinates.

$$Absoloute\ Error = \sqrt{abs(x_t - x_d)^2 + abs(y_t - y_d)^2)} \qquad (1)$$

The error over an image is then calculated from the sum of errors for each corner.

We can use these values to compare each image by looking at the mean error, standard deviation and confidence interval of each method. We can also each method based on the average runtime of each. Another measure of comparison we can use is the number of images in which board detection failed for each method.

The number of failed game board detections allows us to determine the likelihood an average of detection for each method. The mean allows us to visualise which method has the lowest error on average. The standard deviation (SD) of errors shows us how the errors for each method vary from their mean error. The 95% confidence interval (CI) show us a range of values in which we can be certain that the true mean error of each method lies.

SD is calculated as in equation 2, where $\overline{x}$ is the mean and N is the number of data points within the population

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2} \qquad (2)$$

CI is calculated as in equation 3, where Z is 1.96 for a 95% CI.

$$\text{CI} = \overline{X} \pm Z\frac{s}{\sqrt{N}} \qquad (3)$$

5.1.2. Results and Discussion

This section presents, discusses and compares the performance of each method for game board detection. The section ends with an evaluation of the component against the requirements set out in Section 3 .

Method Num.	Board Detection (%)	Mean Error	SD	95% CI	Mean exec. Time (s)
1	94.28%	12.618	13.4	+/- 4.57	2.158
2	82.9%	33.57	61.38	+/- 22.3	2.152
3	42.85%	241.59	361.79	-	0.230
4	91.42%	19.467	24.582	+/- 8.52	2.244
5	97.14%	9.78	7.64	+/- 2.57	2.240
6	100%	13.87	15.69	+/- 5.1	2.515

Table 5 Results of Game board Detection Component

Testing for game board extraction for each edge detection method was carried out as described in 5.1.1. The results produced by these tests can be found summarised in Table 5.

The board was only detected 100% of the time with method 6, however methods 1,4 and 5 produced satisfactory detection rates. Method 3 was strongly affected by lighting and glare within the input image, causing parts of the board to be missed, due to falling outside of the colour thresholds set. Method 2's lower success rate can be accredited to its poor ability to deal with noise – causing extra information to be detected in some cases leading to failures. Method 3's failures where in large part due to images in which boards are placed on surfaces with similar colours (see Figure 24 Error Due to Similar Background Colour for Method 3). With colour being the only descriptor, a wide colour range is needed in order to ensure the board edge is detected properly – making it hard to counteract this problem without more prior knowledge of background.

Figure 24 Error Due to Similar Background Colour for Method 3

The case in which method 5 failed could be determined to be due to an extreme condition with board placement on the edge of a table causing extra points to be recorded (see Figure 25).

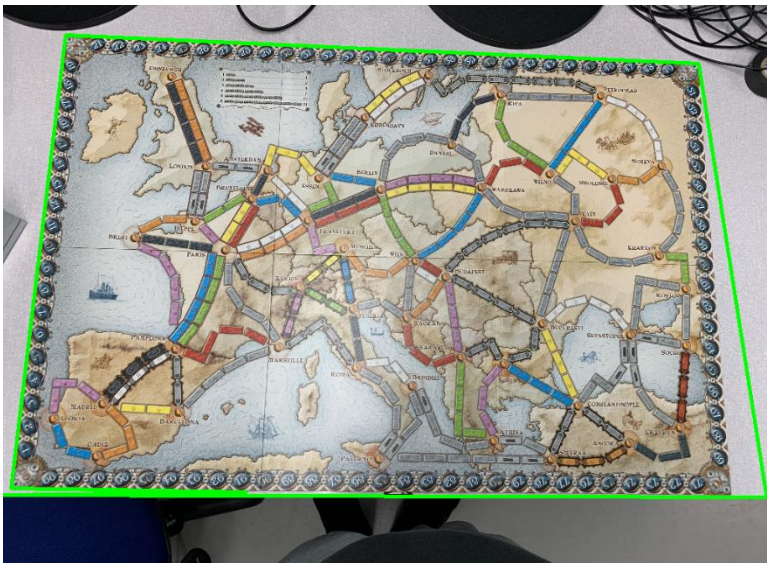


Figure 25 Error Due to Additional Points for Method 5

Overall based on this characteristic alone only methods 2 and 3 can be considered unusable in any such system in their current form. We can also determine that with the addition of colour in combination with shape descriptors, method 6 improves the robustness of the systems rate of detection. The drawbacks of using the colour feature seen in method 3 can be negated with a smaller colour range due to the use of multiple features for game board detection.

In the cases where game boards where detected, looking at the Mean Error (ME) of each method we can see methods 5 of the 6 methods performed very well with low ME values. With methods 3 ME was much larger in comparison, producing coordinates that where barely useable for the next stage of perspective warp. Although method 2 has a relatively low ME of 33.57, its SD of 61.38 with a 95% CI of +/- 22.3. This shows a very large spread of absolute error based on the test images, and hence a lack of reliability of performance. Method 5 showed the greatest accuracy with a ME of 9.78 (95% CI +/- 2.57), confirming the hypothesis of improved performance hypothesised in [18]. However, methods 1, 4 and 6 are not far behind with comparable accuracy. Again methods 2 and 3 can be considered unusable in any such system in their current form based on these results.

The mean execution times for methods 1,2,4 and 5 were all comparable at around 2.2 seconds. Method 3 achieved the best mean execution time at 0.23 seconds, with method 6 (a combination of 3 and 5) executing on average in 2.5 seconds. All of which can be deemed acceptable improving greatly on similar projects – 20 seconds in [34] and 19 seconds in [33].

The ability to meet the requirements of the component set out in 3.2 also bare weight on how well each method performs. Requirement 1.a was met by methods 1,4,5 and 6, all of them identifying the board in greater than 90% of cases. 1.b was again fulfilled by methods 1,4,5 and 6. Finally all methods met 1.c with execution times of less than 2.5 seconds on average. Methods 2 and 3 are do not meet any of the requirements and can therefore again be deemed unusable in any such system.

In conclusion method 6 can be determined to be the best method for edge detection of those outlined. Although ME is slightly higher than that of 5, the reduced number of boards completely missed shows greater robustness to variance in image environment with additional feature descriptors – as hypothesised in 4.3.1.1. The slight increase in execution time is not great enough to cause any significant drawback of method 6 over 5. We can also see that methods based on colour descriptors (method 3) alone are not good enough to produce reliable results within any such implementation with variance in image conditions causing very poor results. The ability of the remaining methods while promising, do not match that of methods 5 and 6. Examples of results for each method can be found in the Appendix section.

5.2. Game Piece Detection Evaluation

5.2.1. Evaluation Method and Metrics

This section discusses the process behind **game piece detection** testing, as well as the metrics used to access performance. The method implemented to test this component used a set of 14 images – containing images both with and without game pieces. These images like in 5.1, varied in a number of ways – gameboard background, perspective and lighting (on or off).

To begin testing we manually counted the number of game pieces within each test image and their location, these will be used to compare to output retrieved from the system to determine accuracy. Upon running each image actual output was compared to expected output for each game tile, recording both the number of correct classifications and the number of extra/missed game pieces determined by the system. Execution time for each image was also recorded.

These values allow us to calculate the mean percentage of correctly classified game tiles, as well as the mean number of missed and extra game pieces to judge performance of the system. SD and CI will also be used as described in 5.1.1.

5.2.2. Results and Discussion

This section presents, discusses and compares the performance of the method for game piece detection using both RGB and HSV colour systems and previous work from 2.4. The section ends with an evaluation of the component against the requirements set out in Section 3.

System	Accuracy	SD	95% CI	Average Num. Errors	Mean exec. Time (s)
This Project: HSV	98%	4.26%	+/- 2.2%	0.64	5.44
This Project: RGB	87%	19.32%	+/- 10.1%	3.64	5.51
[33]	83.65%	-	-	-	100-200

Table 6 Results of Game piece Detection Component

Testing for game piece extraction was carried out as described in 5.1.1. The results produced by these tests can be found summarised in Table 5.

The use of the HSV colour space showed the best accuracy with average accuracy of 98% (95% CI +/- 2.2%) for the correct classification of both empty tiles and those with trains – in comparison the RGB colour space had an average accuracy of 87% (95% CI +/- 10.1%). The HSV colour space also provided a lower average of both false positives and negatives vs the RGB colour space. In conclusion based on accuracy the HSV colour space is noticeably better at retrieving correct player information from the gameboard.

Errors from the component where largely due to using generalised colour ranges being used. This was the case to ensure ease of use on the user – with only one input image being used at any game state. To reduce errors, we could take extra input from the user at game initialisation – creating dynamic colour ranges for each game tile, avoiding performance variance due to the user's environment. This would however limit the user to not change environmental conditions such as lighting throughout each run of the system – decreasing ease of use further.

In comparison methods used in [33] where found to be 83.65% accurate, showing a notable improvement – however the game used contained different game piece types to be found. The mean execution time of my component was also greatly improved, at 5.51 seconds in comparison to 100-200 seconds in [33].

The ability to meet the requirements of the component set out in 3.2 also bare weight on how well it performs. Requirements 2.a and 2.b were met with game pieces of both differing and the same colour as the tiles their placed on being detected with >95% accuracy on average - see Figure 23 and Table 6 Results of Game piece Detection Component. 2.c was also met with an execution time of less than 10 seconds on average at 5.5 seconds. An example of the result for this component for each colour model implemented can be found in the Appendix section.

6. Conclusion

6.1. Project Summary

The main goal of this project was to produce two separate components of a system to automatically score a game of Ticket to Ride. This section details the outcomes that can be ascertained and reviews the performance of the project in terms of the aims that were set out.

The first component, game board detection, involved the development of multiple methods for edge detection. Each method was constructed using various colour and shape feature extraction techniques in combination with different image processing techniques from the literature. The results produced from each method in 5.1.2 are positive in 4 of the methods – using Sobel, Canny and two combinations of multiple features. These 4 methods all met the requirements set out for the component. With the board being identified on average over 90% of the time with a low average absolute error and running within 2.5 seconds. The other two methods, using both Laplace and colour alone, only met timing requirements. We sore that colour descriptors alone, although fast, proved to produce very poor results – however when used in combination with other features can be useful. The results demonstrate that edge detection methods in which multiple features were used produced both the most accurate and robust results. Methods 5 and 6 could both be considered for use within the final system, with the latter being preferred due to its durability against environmental conditions.

The second component, game piece detection was constructed through a combination of image processing and colour feature extraction techniques from the literature. The results produced in 5.2.2 meet the requirements set out for the component. Both game pieces of differing and the same colour as the tiles their placed on are detected with >95% accuracy on average within 5.5 seconds. The cause of errors largely originates from using generalised colour ranges to determine game piece presence. To improve accuracy dynamic colour ranges for each game tile could be calculated through the use of an additional initialisation image. However, this limits the user to not change environmental conditions during the game, adding further restrictions to user experience.

Future work that could be undertaken to extend this project further could make use of artificial intelligence-based approaches as a means to improve the robustness of results - through the implementation of a convolutional neural network for example. This would need the production of a dataset of ladled testing and training images to be developed in order to correctly train the network. To ensure good generalisation (no under/overfitting of testing and training images), we would need there to be a significant amount of data present. If this did indeed improve the robustness of image processing significantly the assumptions made on input images could be relaxed – improving the viability for real world use.

6.2. Statement of Ethics

There are no implications ethically that are brought about by this project. There were both no human participants and personal data used within the project. In addition, all images where produced by me unless explicitly stated otherwise. All data displayed is shown without modifications and was produced by me, the author unless explicitly stated otherwise.

References

[1] G. H. Joblove and D. Greenberg, 'Color space for computer graphics. Computer Graphics', vol. 12, Aug. 1978, pp. 20-25 12. DOI: 10.1145/965139.807362.

[2] ping Tian, Dong, ‘A review on image feature extraction and representation techniques’, International Journal of Multimedia and Ubiquitous Engineering. Jul. 2013, pp. 385-96.

[3] R. Chakravarti and X. Meng, "A Study of Color Histogram Based Image Retrieval, ‘2009 Sixth International Conference on Information Technology: New Generations’, Las Vegas, NV, 2009, pp. 1323-1328. DOI: 10.1109/ITNG.2009.126

[4] M. Tkalcic and J. F. Tasic, ‘Colour spaces: perceptual, historical and applicational background’, The IEEE Region 8 EUROCON 2003. Computer as a Tool., Ljubljana, Slovenia, 2003, pp. 304-308 vol.1. DOI: 10.1109/EURCON.2003.1248032

[5] D. A. Forsyth and J. Ponce, ‘Computer vision: a modern approach’, 2003, pp.21-48. ISBN-10: 0-13-608592-X

[6] P. K. Sahoo, S. Soltani and A.K.C. Wong, ‘A Survey of Thresholding Techniques’, Computer Vision, Graphics, and Image Processing, Volume 41, Issue 2,1988, pp. 233-260, DOI:10.1016/0734-189X(88)90022-9.

[7] N. Otsu, ‘A threshold selection method from gray-level histograms’, IEEE transactions on systems, man, and cybernetics 9, no. 1,1979, pp. 62-66.

[8] D. Zhangand G. Lu, 'Review of shape representation and description techniques', Pattern Recognition, Volume 37, Issue 1,2004, pp. 1-19, DOI:10.1016/j.patcog.2003.07.008.

[9] S.S, Al-amri, N.V. Kalyankar and S.D. Khamitkar, ‘Image Segmentation by Using Thershod Techniques’, computingJournal of Computing, volume 2, issue 5, May 2010

[10] N.R. Pal and S.K. Pal, 'A review on image segmentation techniques', Pattern Recognition, Volume 26, Issue 9, 1993, pp. 1277-1294, DOI:10.1016/0031-3203(93)90135-J

[11] N.M. Zaitoun and M. J. Aqel, 'Survey on Image Segmentation Techniques', Procedia Computer Science, Volume 65, 2015, pp. 797-806, DOI:10.1016/j.procs.2015.09.027.

[12] R. Maini and H. Aggarwal, ‘Study and Comparison of Various Image Edge Detection Techniques’, International journal of image processing (IJIP) 3.1, 2008

[13] M. Sharifi, M. Fathy and M. T. Mahmoudi, ‘A classified and comparative study of edge detection algorithms’, International Conference on Information Technology: Coding and Computing, Las Vegas, NV, USA, 2002, pp. 117-120.

DOI:10.1109/ITCC.2002.1000371

[14] X. Wang, ‘Laplacian Operator-Based Edge Detectors’, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 29, no. 5, pp. 886-890, May 2007. DOI:10.1109/TPAMI.2007.1027

[15] G. Jie and L. Ning, "’An Improved Adaptive Threshold Canny Edge Detection Algorithm’, International Conference on Computer Science and Electronics Engineering, Hangzhou, 2012, pp. 164-168. DOI10.1109/ICCSEE.2012.154

[16] L.S. Hasan, ‘ Evaluate Combined Sobel-Canny Edge Detector for Image Processing’, 2013

[17] M. Fang, G. Yue and Y. Qingcang, ‘The Study on An Application of Otsu Method in Canny Operator’, The 2009 International Symposium on Information Processing, 2009

[18] A. Kalra and R. L. Chhokar, ‘A Hybrid Approach Using Sobel and Canny Operator for Digital Image Edge Detection’, International Conference on Micro-Electronics and Telecommunication Engineering (ICMETE), Ghaziabad, 2016, pp. 305-310. DOI:10.1109/ICMETE.2016.49

[19] S.J. Prince, ‘Computer vision: models, learning and inference’, Cambridge University Press, Jun 2012 ISBN 1107011795

[20] A. Buades, B. Coll, and J.M. Morel, ‘Non-Local Means Denoising’, Image Processing On Line, 2012, pp. 208–212, DOI:10.5201/ipol.2011.bcm_nlm

[21] A. Buades, B. Coll and J.M. Morel, ‘A non-local algorithm for image denoising’, 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 2005, pp. 60-65 vol. 2, DOI:10.1109/CVPR.2005.38

[22] R. C. Gonzalez and R. E. Woods, ‘Digital Image Processing’, Mar 2002

[23] M. Nixon and A.S. Aguado, ‘Feature Extraction and Image Processing for Computer Vision’, 2012

[24] N. Dhanachandra, K. Manglem and Y.J. Chanu, ‘Image segmentation using K-means clustering algorithm and subtractive clustering algorithm’, Procedia Computer Science 54, 2015, pp. 764-771.

[25] A. Zucconi, ‘How to find the main colours in an image’, May 2015, Accessed on: 24 Apr. 2019, [Online]. Available: https://www.alanzucconi.com/2015/05/24/how-to-find-the-main-colours-in-an-image/

[26] S. Suzuki, KeiichiA be, 'Topological structural analysis of digitized binary images by border following', Computer Vision, Graphics and image Processing, Volume 30, Issue 1, 1985, pp. 32-46, DOI:10.1016/0734-189X(85)90016-7.

[27] Wikipedia, The Free Encyclopedia, ‘Ramer–Douglas–Peucker algorithm’ [Online]. Available: https://en.wikipedia.org/wiki/Ramer%E2%80%93Douglas%E2%80%93Peucker_algorithm

[28] Intel, “OpenCV Documentation“, June 2000 [Online] Available: https://docs.opencv.org/3.4.6/index.html Accessed on: Apr. 29, 2019.

[29] Travis Oliphant, “NumPy“, 1995 [Online] Available: https://www.numpy.org/ Accessed on: Apr. 29, 2019.

[30] David Cournapeau, “Scikit-Learn“, June 2007 [Online] Available: https://scikit-learn.org/ Accessed on: Apr. 29, 2019.

[31] Guido van Rossum, “Python“, 1990 [Online] Available: https://www.python.org/ Accessed on: Apr. 29, 2019.

[32] S. Scher, R. Crabb and J. Davis, “Making Real Games Virtual: Tracking Board Game Pieces”, 2009 DOI:10.1109/ICPR.2008.4761307.

[33] D. Lang, “’Agricola’ Board Game Assist Program”, EGGN 512-Computer Vision, May, 2012
Available:https://pdfs.semanticscholar.org/e566/d332d1c62a2c4a2db4158c16bc93b77975bb.pdf

[34] T. Musil, “Optical Game Position Recognition in the Board Game of Go”, Bachelor thesis, Charles University, Department of Applied Mathematics, Prague, July 2014 Available: http://tomasm.cz/imago_files/go_image_recognition.pdf

[35] A. K. Seewald, “Automatic Extraction of Go Game Positions from Images: A Multi-Strategical Approach to Constrained Multi-Object Recognition”, Vienna, Austria, Dec. 2009 Available: https://www.seewald.at/files/2007-04.pdf

[36] Y. Liu, S. Liu, Y. Cao and Z. Wang, "A practical algorithm for automatic chessboard corner detection," 2014 IEEE International Conference on Image Processing (ICIP), Paris, 2014, DOI: 10.1109/ICIP.2014.7025701

[37] A. M. Czyzewski, “An Extremely Efficient Chess-board Detection for Non-trivial Photos.”, CoRR, 2017 Available: https://arxiv.org/pdf/1708.03898.pdf

[38] K. Y. Tam, J. A. Lay and D. Levy, "Automatic Grid Segmentation of Populated Chessboard Taken at a Lower Angle View," 2008 Digital Image Computing: Techniques and Applications, Canberra, ACT, 2008 DOI:10.1109/DICTA.2008.40

[39] J. Seo, et al, “Fast Contour-Tracing Algorithm Based on a Pixel-Following Method for Image Sensors”, Mar. 2016 DOI:10.3390/s16030353

Appendix

Figures 26–31 Show an example of component 1's output for methods 1-6 on the image within Figure 11 Ticket to Ride Europe game board before perspective warp and contour finding.

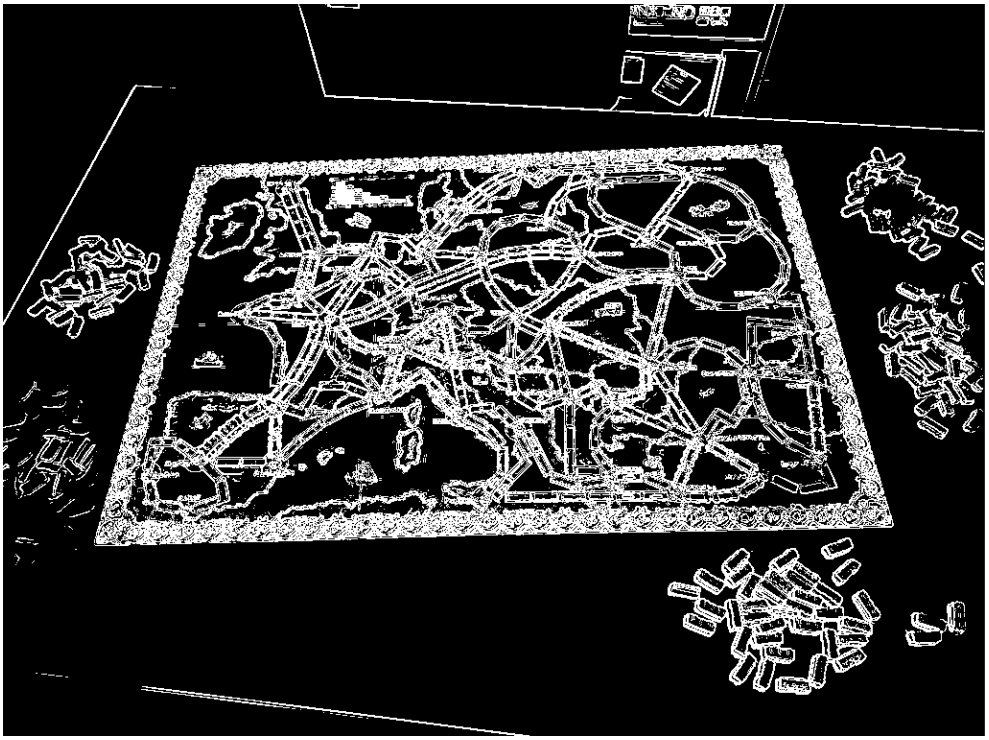


Figure 26 Method 1 Output

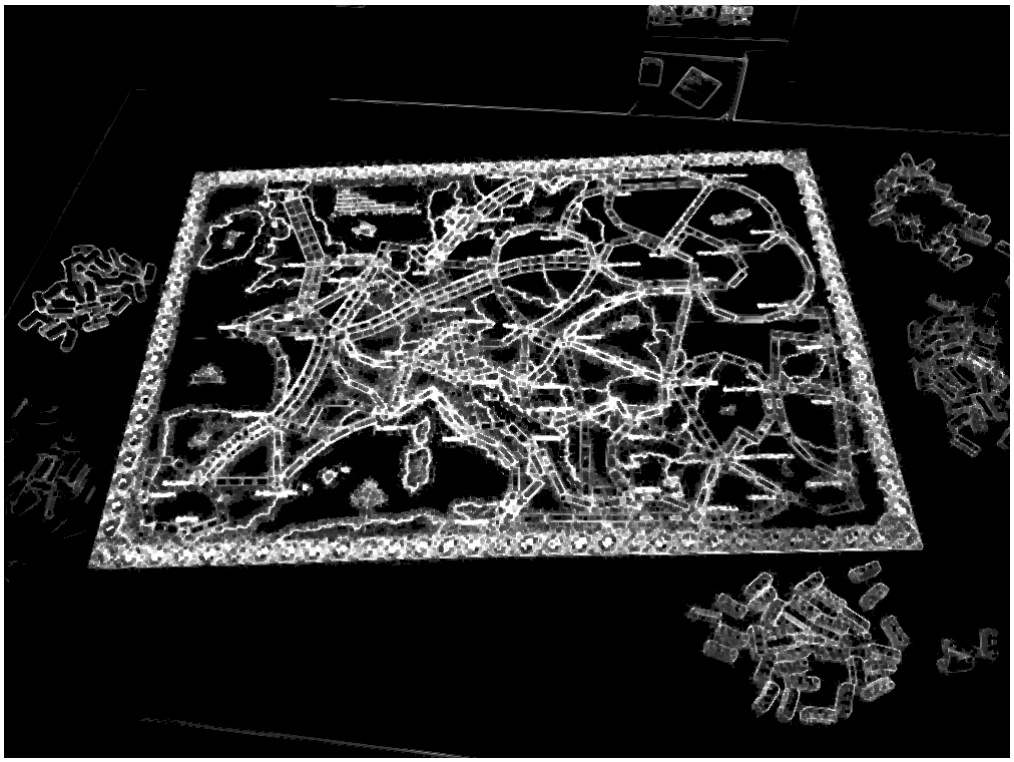


Figure 27 Method 2 Output

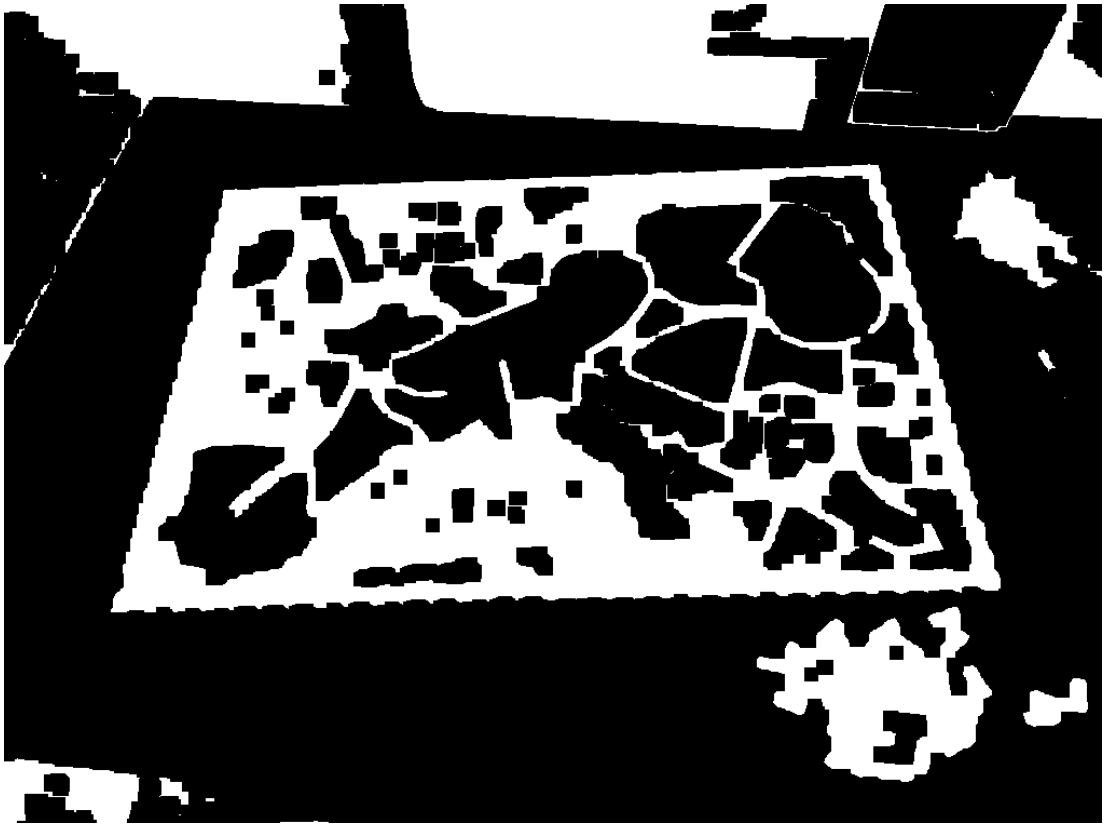


Figure 28 Method 3 Output

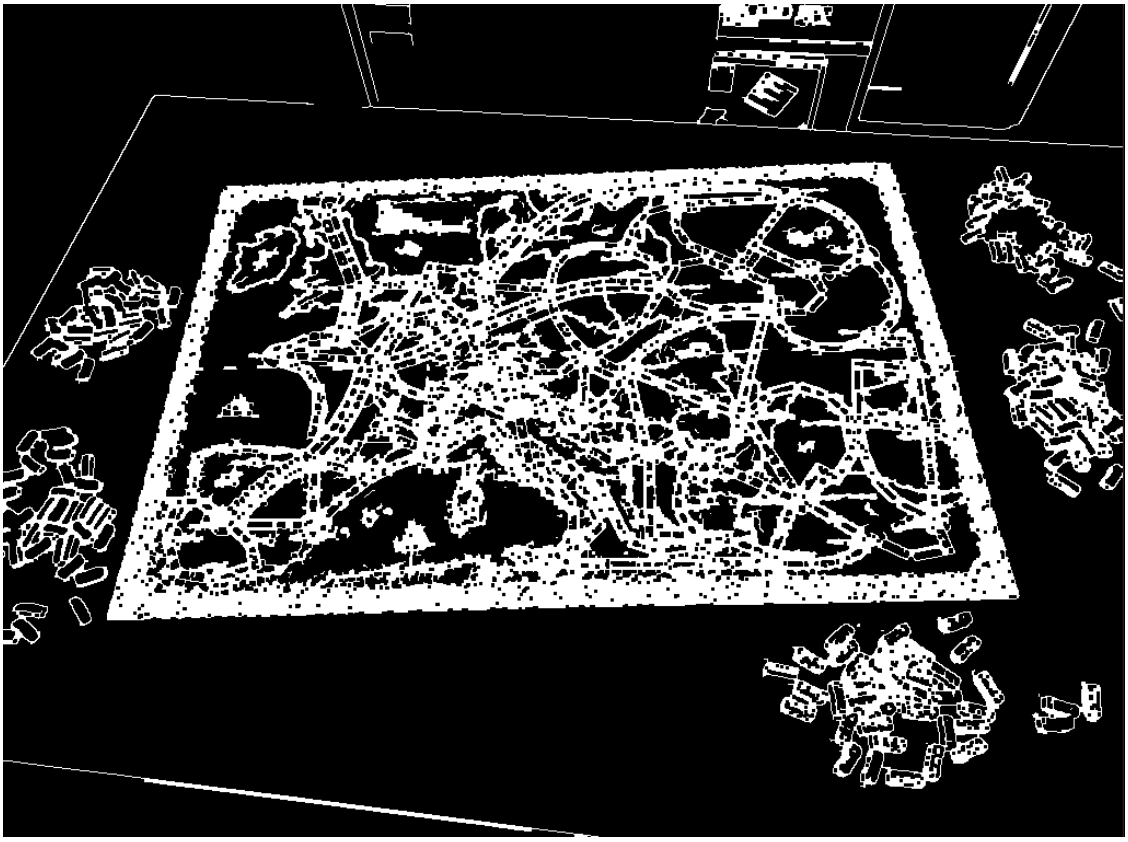


Figure 29 Method 4 Output

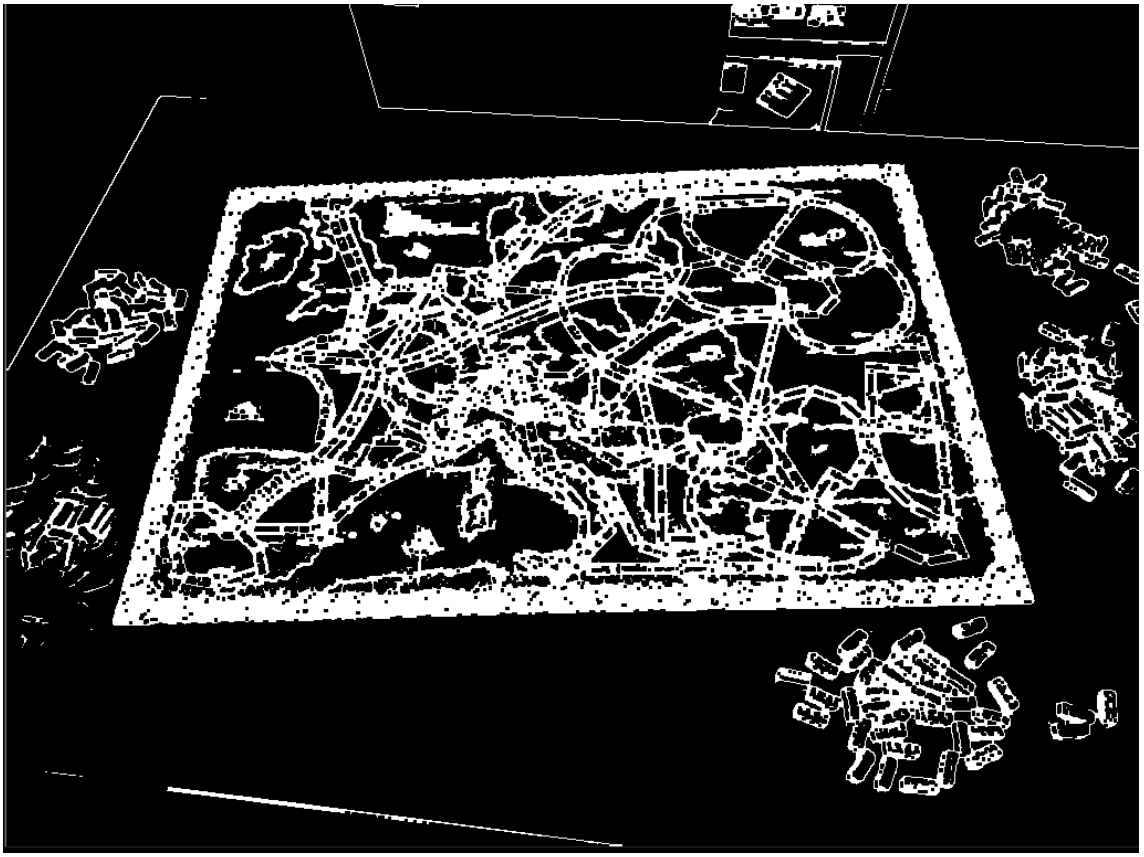


Figure 30 Method 5 Output

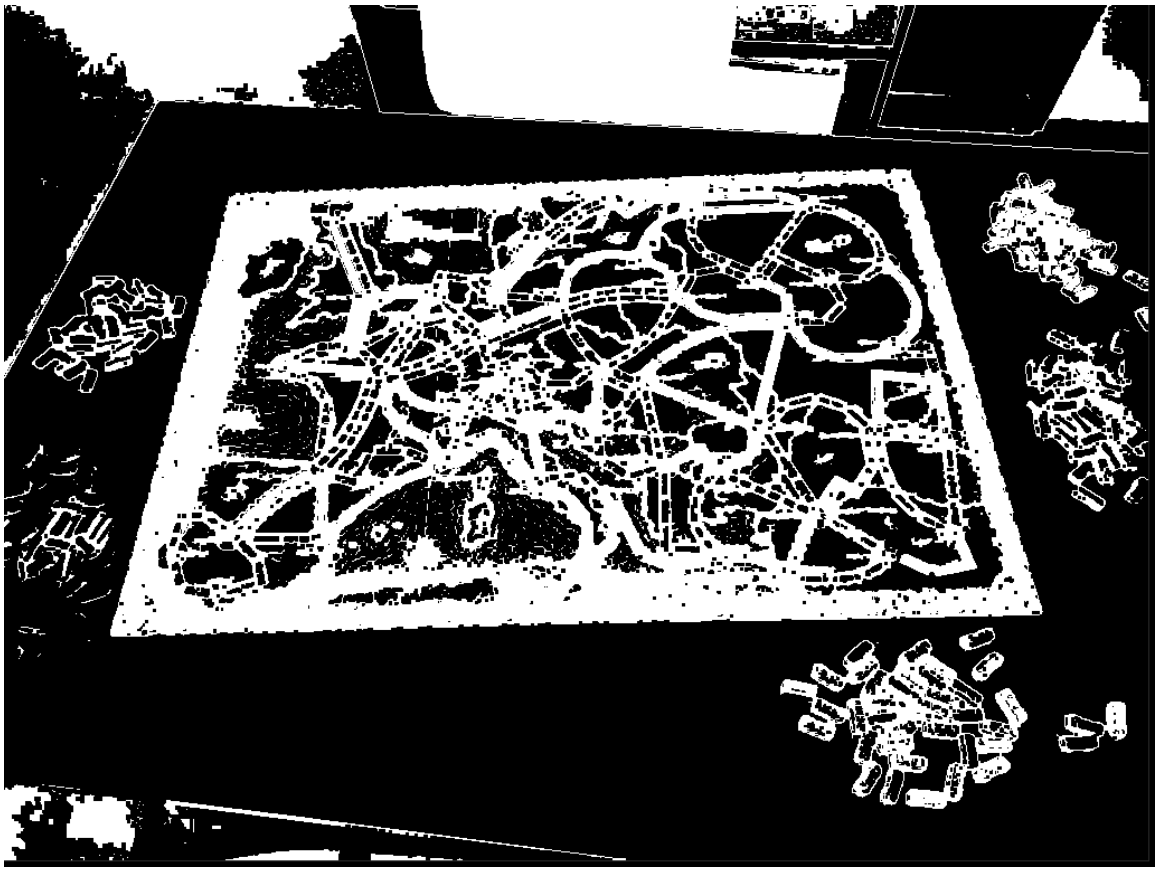


Figure 31 Method 6 Output

Figures 32–33 Shows an example of the component 2 output for methods RGB and HSV.

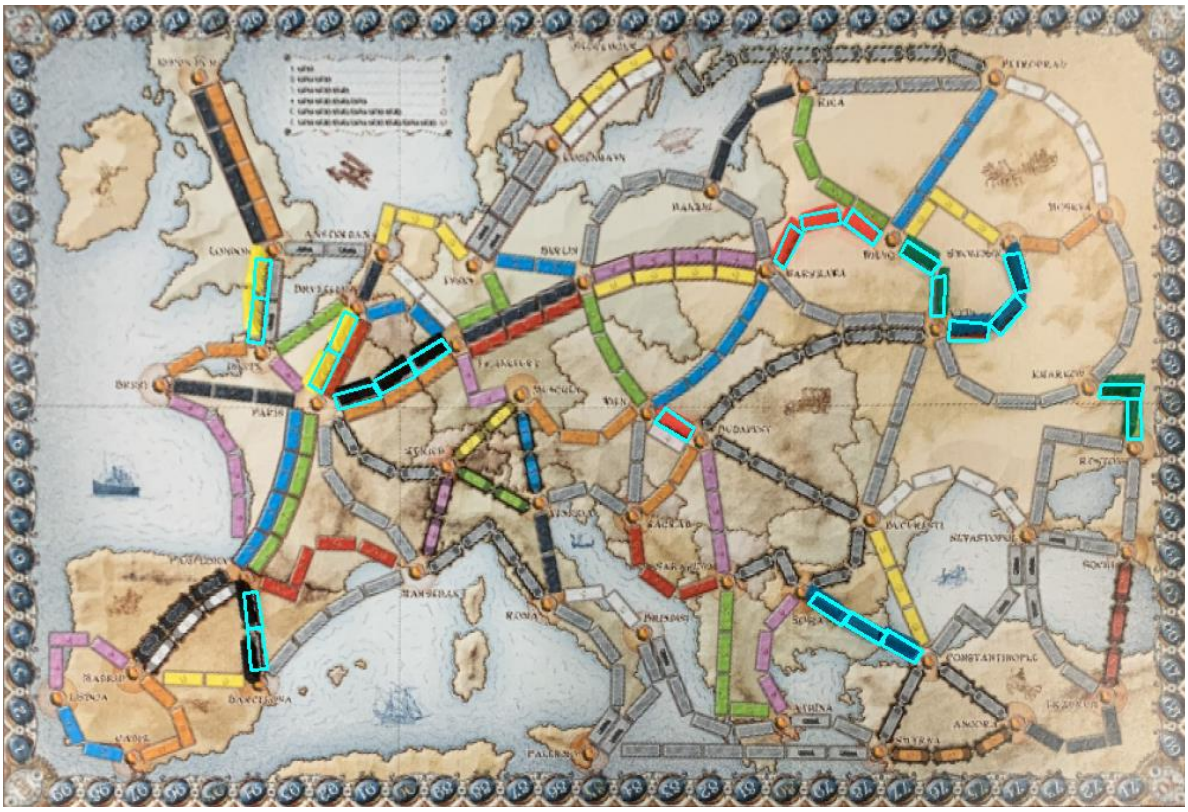


Figure 327 Using HSV

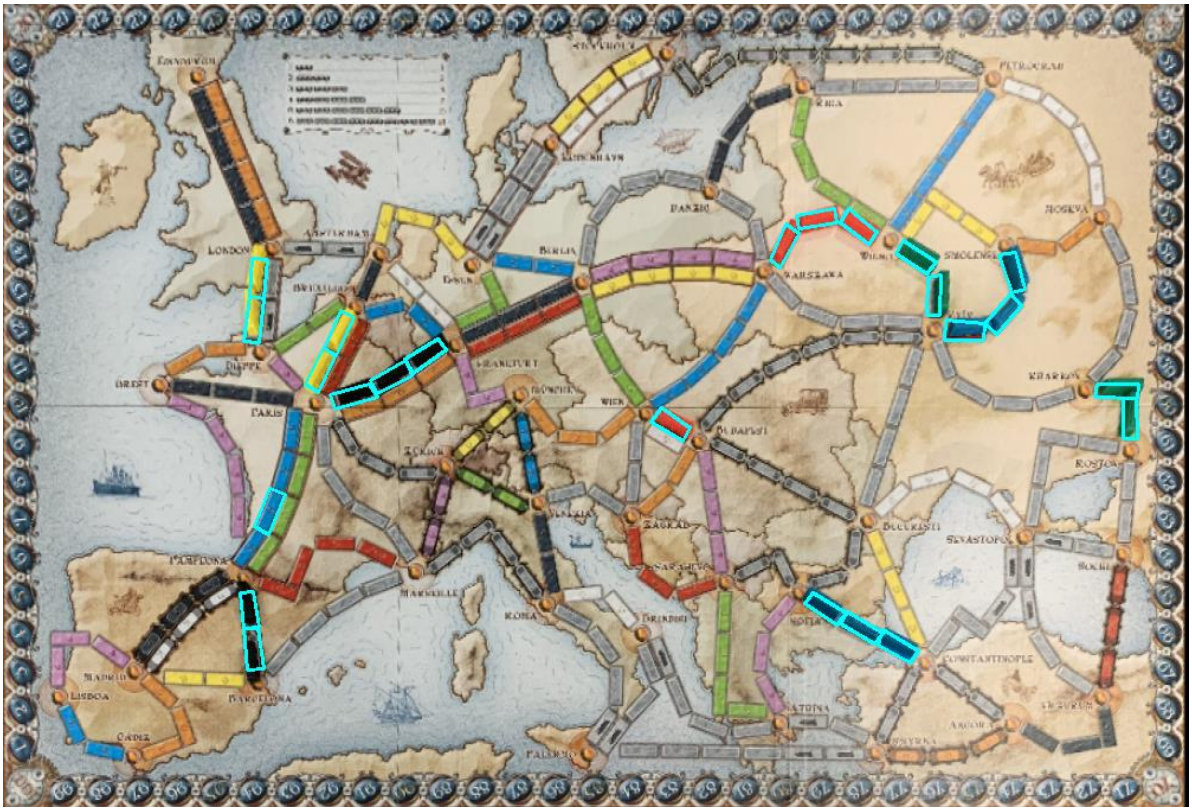


Figure 33 Using RGB