DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

AN ARTIFICIAL INTELLIGENCE APPLICATION WITH IMAGE RECOGNITION

by
Lütfi MUTLU

September, 2021
İZMİR

AN ARTIFICIAL INTELLIGENCE APPLICATION WITH IMAGE RECOGNITION

A Thesis Submitted to the
Graduate School of Natural and Applied Science of Dokuz Eylul University
In partial Fulfillment of the Requirements for the Degree of Master of Science
in Computer Science, Computer Science Program

by
Lütfi MUTLU

September, 2021
İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"AN ARTIFICIAL INTELLIGENCE APPLICATION WITH IMAGE RECOGNITION"** completed by **LÜTFİ MUTLU** under supervision of **PROF. DR. EFENDİ NASİBOĞLU** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

……...

Prof. Dr. Efendi NASİBOĞLU

_____

Supervisor

…… ……

Asst. Prof. Dr. Gazihan ALANKUŞ Assoc. Prof. Dr. Fidan NURİYEVA

_____ _____

(Jury Member) (Jury Member)

_____

Prof. Dr. Özgür ÖZÇELİK

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGEMENTS

I would like to thank my supervisor Prof. Dr. Efendi NASİBOĞLU for his support, suggestions and guidance throughout the whole study. I also want to thank my family for their unlimited patience throughout this project and my entire life.

Lütfi MUTLU

AN ARTIFICIAL INTELLIGENCE APPLICATION WITH IMAGE RECOGNITION

ABSTRACT

In this thesis, firstly the base program is developed for implementing and testing image processing algorithms. Image loading (from hard drive and webcam), image slicing and mixing functions are implemented. Puzzle solving algorithm based on one side edge detection with candidate adjacent cells is developed, tested and analysed with different images. As a consequence of this, the base of a new algorithm based on grouping jigsaw pieces (like Tetris game pieces) is applied. Width and height limit conditions are added according to the number of pieces in the puzzles. It has been provided to make multiple solutions automatically to increase the solution performance for choosing the optimal solution. On the second stage, pieces with lowest match scores are swapped among themselves for better solutions. The program consists of Tetris Mode and Unrotated Pieces Mode. In the Tetris Mode, pieces with similarity below the threshold are grouped together. These groups are displayed in the Solutions Window on the GUI. In the Unrotated Pieces Mode, the solutions and their scores are displayed here as images. Simulation of pieces, displaying empty/candidate spaces, puzzle boundaries and displaying the solutions in the Solutions Window are carried out at the same time, during the solution process.

Keywords: Jigsaw puzzle, simulation, edge matching, image recognition.

GÖRÜNTÜ TANIMA ILE BIR YAPAY ZEKA UYGULAMASI

ÖZ

Bu tezde, görüntü işleme algoritmalarını uygulamak ve testlerini yapmak için öncelikle temel bir program geliştirilmiştir. Görüntü yükleme (sabit sürücüden ve internet kamerasından), görüntü dilimleme ve karıştırma işlevleri uygulanmıştır. Komşu aday hücrelerle tek taraflı kenar algılamaya dayalı bulmaca çözme algoritması geliştirilmiş, test edilmiş ve farklı görüntülerle analiz edilmiştir. Bunun bir sonucu olarak, yapboz parçalarının (Tetris oyun parçaları gibi) gruplandırılmasına dayalı yeni bir algoritmanın temeli uygulanmıştır. Bulmacalardaki parça sayısına göre genişlik ve yükseklik sınırı koşulları eklenmiştir. En uygun çözümün bulunması ve çözüm performansını artırmak için birden fazla çözümün otomatik olarak yapılması sağlanmıştır. İkinci aşamada, en düşük eşleşme puanına sahip parçalar, daha iyi çözümler için kendi aralarında değiştirilmiştir. Program, Tetris Modu ve Döndürülmemiş Parçalar Modu olmak üzere iki bölümden oluşur. Tetris Modunda, belirli bir eşiğin altında benzerliğe sahip parçalar birlikte gruplandırılır. Bu gruplar, kullanıcı arayüzünde sağ tarafdaki Çözümler Penceresinde görüntülenir. Döndürülmemiş Parçalar Modunda, çözümler ve puanlar da burada resim olarak görüntülenir. Çözüm sürecinde parçaların simülasyonu, boş/aday boşlukların gösterilmesi, bulmaca sınırlarının gösterilmesi ve Çözümler Penceresinde çözümlerin gösterilmesi aynı anda gerçekleştirilir.

Anahtar kelimeler: Yapboz, simülasyon, kenar eşleme, görüntü tanıma.

CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

CHAPTER ONE

INTRODUCTION

1.1 Overview

In daily life standard jigsaw puzzles are solved by many people young and old alike. The most known method is clustering the pieces by their colours and shapes for searching the required pieces faster. Other most known jigsaw puzzle solving method is starting from edges because they have more certain shapes that can be more easily recognised. So, it provides information about contiguous neighbours that reduces search spaces and defines the borders for other pieces need to be fitted. Sometimes it is possible to put a puzzle piece in the wrong place and it couldn't be recognised until a deadlock. Then, a trial-and-error strategy is applied for other possible pieces and places.

Sometimes jigsaw puzzles can be solved more easily by computers if the puzzle pieces have colour relations and sometimes can be solved by human brains if the puzzles have certain shapes and figures.

Standard puzzles are made by cutting images printed on hard surfaces into patterns of interlocking pieces. With the pieces falling apart and randomly shuffled, the real challenge is to put those pieces back together in the original painting. The difficulty of the problem is determined by the number of parts, the shape of the parts, and the graphical composition of the pictures. Common rules are generally followed when producing these puzzles to achieve a desired level of complexity. Most of the pieces used in recreational jigsaw puzzles have a rectangular outer border and when put together form a rectangular grid where each interior piece interlocks perfectly with its adjacent neighbours (Goldberg et al.,2002). Puzzles that conform to these rules are sometimes called as canonical jigsaw puzzles (Yao and Shao, 2003).

1.2 Digital Image File Formats

Image file formats are standardized tools for editing and storing digital images. We need to know these formats for editing and image processing operations and we can also convert these file formats. Image files consist of digital data in one of these formats that can be rasterized for use on a computer screen or printer. An image file format can store data in uncompressed, compressed, or vector formats. When rasterized, an image becomes a grid of pixels, each containing a series of bits to determine its colour equal to the colour depth of the device displaying it. Main five digital image file formats are given below (Caroline, 2019).

1.2.1 TIFF/TIF file format

TIFF stands for Tagged Image File Format and create very large file sizes. TIFF images are uncompressed and therefore contain a lot of detailed image data. TIFFs are also extremely flexible in terms of colour (can be grayscale or CMYK for print or RGB for web) and content. TIFF is the most common file type used in photography software (like Photoshop) and page layout software (like Quark and InDesign), because again a TIFF contains a lot of image data (Wikipedia, 2019).

1.2.2 JPEG/JPG file format

JPEG stands for Joint Photographic Experts Group, which created this standard for this type of image formatting. JPEG files are compressed images to store a large amount of information in a small file. Most digital cameras store photos in JPEG format because then you can take more photos on one camera card than you can in other formats. A JPEG is compressed in such a way that it loses some image detail during compression to make the file smaller. JPEG files are often used for photos on the web because they create a small file that is easily loaded onto a web page and looks good as well (Wikipedia, 2019).

1.2.3 GIF file format

GIF stands for Graphics Interchange Format. This format compresses images, but unlike JPEG, the compression is lossless (no detail is lost in compression, but the file cannot be made as small as JPEG). GIFs also have an extremely limited colour range that is suitable for the web but not for print. This format is never used for photography due to the limited number of colours (Wikipedia, 2019).

1.2.4 PNG file format

PNG stands for Portable Network Graphics. It was created as an open format to replace GIF because GIF was patented by a company and no one else was willing to pay license fees. It also provides a full colour gamut and better compression. Used almost exclusively for web images, never for print images. For photos, PNG is not as good as JPEG as it creates a larger file (Wikipedia, 2019).

1.2.5 Raw image files

Raw image files usually contain data from a digital camera. The files are called raw because they are not processed and therefore cannot be edited or printed yet. There are many different raw formats. Each camera company usually has its own proprietary format. Raw files usually contain large amounts of uncompressed data. Therefore, large file size due to the minimal to no compression. Usually, they are converted to TIFF before editing and colour correction operations. (Wikipedia, 2019).

Raw image files need much more operation for image processing operations and they can be used for different purposes. Raw images can be processed by digital cameras or some special programs on computers. Some features like saturation, shadow details, white balance, contrast etc. can be adjust with raw images.

1.3 RGB Colour Model

RGB stands for red, green, and blue. It refers to a system of representing colours to be used on a computer screen. Red, green, and blue can be combined in various proportions to produce any colour in the visible spectrum. Red, green and blue levels are represented by a decimal range equivalent to a binary number range from 0 to 255. The total number of available colours is 256x256x256 or 16,777,216 possible colours (WhatIs.com, 2019). For example: To obtain the exact red colour, level of red must be 255 while green and blue levels are 0. If all colour levels are 255 then we obtain white colour. Inversely, if all colour levels are 0, then we obtain black colour.

The way computers store, manipulate and display graphics dates back to the early television technology that was first used to create computer displays. Billmeyer and Saltzman (2000) examine this area including decomposition of colour in detail.

The RGB colour model is used for sensing, representing and displaying images in electronic systems such as monitors and televisions, although it has also can be used in conventional photography. Before the electronic age, there was a solid theory behind the RGB colour model shown in Figure 1.1, based on human perception of colour (Wikipedia, 2019).

Figure 1.1 Additive colour mixing (Wikipedia, 2019)

Therefore, an RGB value does not define the same colour across devices without some form of colour management. Adding blue to red gives purple or adding green to red gives yellow or adding blue to green gives cyan. Mixture of all red, green and blue gives white (Wikipedia, 2019).

Figure 1.2. RGB pixels in a television (on the left) and RGB phosphor dots in a monitor (on the right) (Wikipedia, 2019)

RGB colour model is also used in the thesis for reading RGB values of each pixel on the edges of the jigsaw puzzle pieces. GetPixel command (Bitmap.GetPixel(x position, y position)) is used for reading and SetPixel command (Bitmap.GetPixel(x position, y position, colour)) is used for changing the RGB values of pixels in C Sharp programming language. Special colours can be created with Color.FromArgb command (Color.FromArgb(Alpha Value, Red Value, Green Value, Blue Value)) by using RGB colour model. Transparency of a colour or pixel can be also adjusted with alpha value.

CHAPTER TWO

JIGSAW PUZZLE SOLVING ALGORITHMS

2.1 Stochastic (ST) Algorithms

Stochastic algorithms are methods that generate and use random variables. For stochastic problems, random variables appear in the formulation of the optimization problem, involving arbitrary objective functions or arbitrary constraints. These algorithms are designed for problems with natural random noise or deterministic problems solved by injected randomness.

2.2 Structured Stochastic (SS) Algorithms

The structured stochastics algorithms enable an additional procedure that rotates each piece of a randomly generated arrangement. After each rotation, the quality of the solution is re-evaluated and all improvements are accepted. It is important to note that the sequential rotation of pieces leads to a small reduction in the size of the relevant search space. A single pass of this optimization method ensures that an initially generated arrangement containing only a few pieces of incorrect orientation can be improved upon (Tybon, 2004).

2.3 Genetic Algorithms (GA)

The genetic algorithm (GA) is a meta-heuristic inspired by the natural selection process belonging to the larger class of evolutionary algorithms (EA). Genetic algorithms are widely used to generate high-quality solutions for optimization and search problems, relying on bio-inspired operators such as crossover, mutation and selection (Hatti, 2018).

2.4 Simulated Annealing (SA) Algorithms

Simulated Annealing is a probabilistic technique for approximating the global optimum of a given function. Particularly, it is a meta-heuristic to approach global optimization in a large search space for an optimization problem. Usually used when the search area is separate. Simulated annealing algorithms may be preferred to alternatives such as gradient descent (Wikipedia, 2019).

2.5 Tabu Search (TS) Algorithm

Tabu search is a metaheuristic search method that uses local search methods used for mathematical optimization. Local searches bring up a possible solution to a problem and check near neighbours in hopes of finding a better solution. Taboo search improves the performance of local search by relaxing the ground rule. First, moves worsening at each step are acceptable if no improvement moves are available (Roy et al., 2019).

2.6 Constraint Satisfaction (CS) Programming

Constraint satisfaction in artificial intelligence (AI) and operations research is the process of finding solutions to a set of constraints that impose conditions that variables must meet. So, a solution is a set of values for variables that satisfy all constraints, a point in the applicable region. The techniques used in constraint satisfaction depend on the type of constraints being considered. Constraint propagation is another method used in this type of problem; most are generally incomplete, meaning they can fix the problem or prove unsatisfactory, but not always (Davenport and Tsang, 1999).

Constraint Satisfaction Programming (CSP) is the youngest of all the solutions algorithm techniques devised for solving NP-C problems. The theoretical framework for CSP (Goldberg et al., 2002), involves searching for values for problem variables from finite domains, subject to the satisfaction of a set of constraints applied to combinations of domain values.

The comparison of above six algorithms for solving jigsaw puzzles is made by Robert Tybon in his Ph. D. thesis (Tybon, 2004) shown in Table 2.1 and proves that Constraint Satisfaction (CS) Programming is better from other algorithms mentioned above.

Table 2.1 Times taken (in milliseconds) to find a valid solution for different puzzle sizes by all implemented algorithms (Tybon, 2004)

Puzzle Size	ST	SS	GA	SA	TS	CS
2x2	406	1	1	1	1	1
2x3	1017250	422	31	15	1	1
2x4	not available	1906	141	47	16	1
3x3	not available	10406	219	63	16	1
2x5	not available	149531	1907	547	16	1
3x4	not available	not available	9831	1531	297	1
3x5	not available	not available	53188	3718	453	15
4x4	not available	not available	111453	3985	672	15
4x5	not available	not available	2326281	70688	21297	16

CHAPTER THREE

THE PROPOSED APPROACH

3.1 The Proposed Approach

In this work, a new and different method has been tried by considering the existing literature with simulation and webcam features. The proposed approach is developed on the basis of the brute force algorithm.

The most common image formats like PNG, JPG, JPEG and BMP can be used with the program or new image can be taken from the webcam.

To determine the similarity of two puzzle pieces, RGB decomposition must used to compare the numeric values assigned to each edge of the puzzle pieces. The technique developed here is similar to one used with success by Glassner (2002) in the reconstruction of shredded documents.

Width and height limit conditions are added according to the number of pieces in the puzzles. For this, the minimum and maximum placement positions of X and Y of the pieces placed in the work area are taken into account. As soon as the width or height requirement is met, width or height limits are set. In this way, the solution was kept in puzzle dimensions. This feature can be used on puzzles of known size. It can be also deactivated in the program, if we have the puzzles of unknown size.

The solution performance of the puzzles varied according to the first piece placed randomly. Therefore, it has been provided to make multiple solutions automatically to increase the solution performance. The score of each solution was noted and the record was constantly updated. Each time the record is updated, the image of the record solution temporarily kept in memory is also updated. The number of trials can be adjusted from the GUI. When the specified number of trials is reached, the best solution obtained in these trials is displayed with the "Show the Best Solution" button. In addition, the scores of each solution and the best solution are also written

on the images. In this way, it can be compared. Starting piece of the solution is also shown with red dots and red frames are added on each solution image in Solution Window (Figure 3.1). The best scores are also listed with a list box.

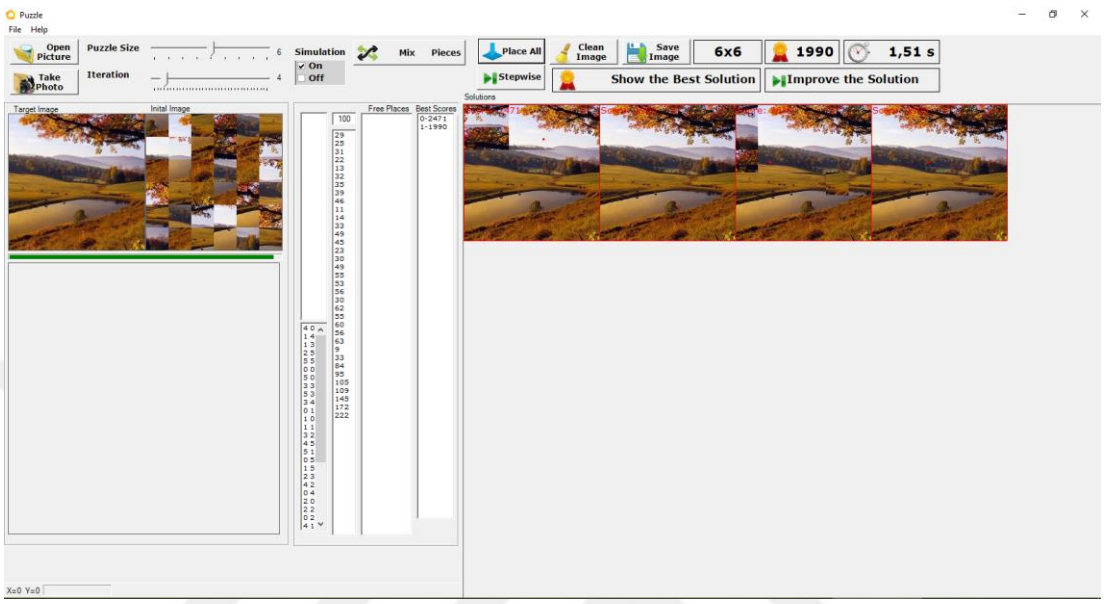

Figure 3.1 The main program GUI

The program currently includes Tetris Mode and Unrotated Pieces Mode. In the Unrotated Pieces Mode, all pieces are combined in a single image. In the Tetris Mode, pieces with similarity below the threshold are grouped together. These groups are displayed in the Solutions Window on the right side of the GUI. In the Unrotated Pieces Mode, the solutions and their scores are displayed here as images.

During the solution process, simulation of pieces, empty spaces and puzzle boundaries (displaying as a images) and displaying the solutions in the Solutions Window at the same time causes a significant waste of time. Thus, the Simulation On-Off option provides faster solutions without simulations.

Simulation option is added to existing program as shown in Figure 3.2. On the sample simulation, placed pieces are shown with their colours, suitable pieces for candidate pieces are shown with green colours (green cells) and borders are shown with red colours (red cells) on the Figure 3.2. There is only two height borders, one line for top and one line for bottom border. Width borders aren't found yet in this

example. Because new pieces can be placed on the left side or right side of placed pieces.

Figure 3.2 Simulation On-Off Option and simulation of placed pieces, suitable pieces (with green) and borders (with red)

3.2 Compatibility of Jigsaw Puzzle Pieces

In this work, for computing the compatibility value, $C(P_0, P_{cand}, d)$ between two jigsaw puzzle pieces where P_0 is placed piece and P_{cand} is the candidate piece and d is the direction of merging edge.

- if d=0: P_{cand} is on the right side of P_0,
- if d=1: P_{cand} is on the bottom side of P_0,
- if d=2: P_{cand} is on the left side of P_0 and
- if d=3: P_{cand} is on the top side of P_0.

Firstly, the absolute differences of the colour gradients on the merging edges of P_0 and P_{cand} is calculated. We define the absolute difference $D(E_0, E_{cand}, RGB)$ with 3 columns (RGB: red, green and blue colour channels), and E rows (where E is the jigsaw pieces' edge length as pixels, E_0: candidate edge of placed piece, E_{cand}: candidate edge of candidate piece).

We define the absolute colour difference of two edges E_0 and E_{cand} as follows (where p_n is nth pixel of the edges E_0 and E_{cand}, p_0: placed pixel of E_0 and p_{cand}: candidate pixel of E_{cand}):

$$D(E_0, E_{cand}, RGB)=[D(E_0, E_{cand}, Red) + D(E_0, E_{cand}, Green) + D(E_0, E_{cand}, Blue)]/3 \quad (3.1)$$

$$D(E_0, E_{cand}, Red) = \frac{1}{P}\sum_{n=1}^{P}\left|p_{0_n}(Red) - p_{cand_n}(Red)\right| \quad (3.2)$$

$$D(E_0, E_{cand}, Green) = \frac{1}{P}\sum_{n=1}^{P}\left|p_{0_n}(Green) - p_{cand_n}(Green)\right| \quad (3.3)$$

$$D(E_0, E_{cand}, Blue) = \frac{1}{P}\sum_{n=1}^{P}\left|p_{0_n}(Red) - p_{cand_n}(Blue)\right| \quad (3.4)$$

The compatibility value $C(P_0, P_{cand}, d)$ of two jigsaw puzzle pieces P_0 and P_{cand} on the direction of d is found as:

$$C(P_0, P_{cand}, d) = D(E_0, E_{cand}, RGB) \quad (3.5)$$

We define a jigsaw puzzle dimensions as m x n. For a square shaped jigsaw puzzle, $m = n$ and the count of pieces, $c = m^2$. Thus, before starting to solve a jigsaw puzzle we have an array list of jigsaw puzzle pieces with the dimension c. We define this array list as L_{cand} and the array list for placed jigsaw pieces as L_{placed} and the array list which shows possible free places for candidate jigsaw pieces with directions as L_{free} (with dimension s). In every step of the jigsaw puzzle solving process, the minimum compatibility value $C_{min}(P_0, P_{cand}, d)$ is found as follows:

```
while c>0 do
  for j=0 to s do
        for i=0 to c do
              value=C(Lfree(j), Lcand(i), d(i))
              if value<min then
                    min=value
                    Lfreemin=Lfree(j)
                    Lcandmin=Lcand(i)
                    dmin=d(i)
               end if
        end for
  end for
  Place Lcandmin on Final Image(Lfreemin)
  Add new candidate adjacent places of Lfreemin to Lfree
  Delete old adjacent places from Lfree
  Update s
  Add Lcandmin to Lplaced
  Delete Lcandmin from Lcand
  c=c-1
end while
```

Figure 3.3 Pseudo code for the first step

3.3 First Step for Solving the Jigsaw Puzzle

The Jigsaw Puzzle Application is developed with C# code. Firstly, an image file is opened with file browser or taken from the webcam. Then it is separeted to pieces and the number of pieces are added to a list box. The number of pieces can be set with a trackbar. After slicing, the pieces are mixed in the list box and new mixture is showed on the program interface.

Now the jigsaw puzzle is ready to be solved. The operations mentioned below is applied in order to solve the jigsaw puzzle.

1. First piece of jigsaw puzzle image in the first list which shows image pieces that will be placed, is placed on the final image and removed from

first list and added to second list which shows image pieces that have been placed.

2. Possible adjecent places are added to third list and painted with palegreen on the final image. The directions of each possible places are also recorded for faster edge matching operations.

3. Each pieces in the first list are placed on the final image by considering the possible places in third list. For each placement and edge matching direction, highest matching score is obtained. Then the best matching image piece is placed on the final image. New possible adjecent places are added to third list and painted on the final image.

4. Repeat (3) until the first list (images will be placed) is empty.

Matching score of each image couple is obtained by summing RGB values on the couple images' matching edges (pixels).

Example 1:

An image is taken from the webcam and sliced as 6x6 pieces as shown in Figure 3.4. Then all the pieces are mixed several times. Final solution of the jigsaw puzzle is obtained by applying the algorithm mention above. The final image with best solution score (1328 in this example) is shown on the left down side. Other ten solutions are also shown on Solution Window on the right side. Solutions are also listed with scores and starting pieces are listed in the Table 3.1.

Table 3.1 The scores of the solutions for Example 1 with the optimum solution

Solution Number	Starting Piece Coordinate	Score (Lower is Better)	
1	1 -0	1724	
2	8 -8	1510	
3	**6 -5**	**1328**	(The Optimum Solution)
4	2 -8	1370	
5	8 -2	1370	
6	0 -6	1603	
7	2 -1	1472	
8	3 -8	1370	
9	0 -6	1370	
10	7 -0	1641	

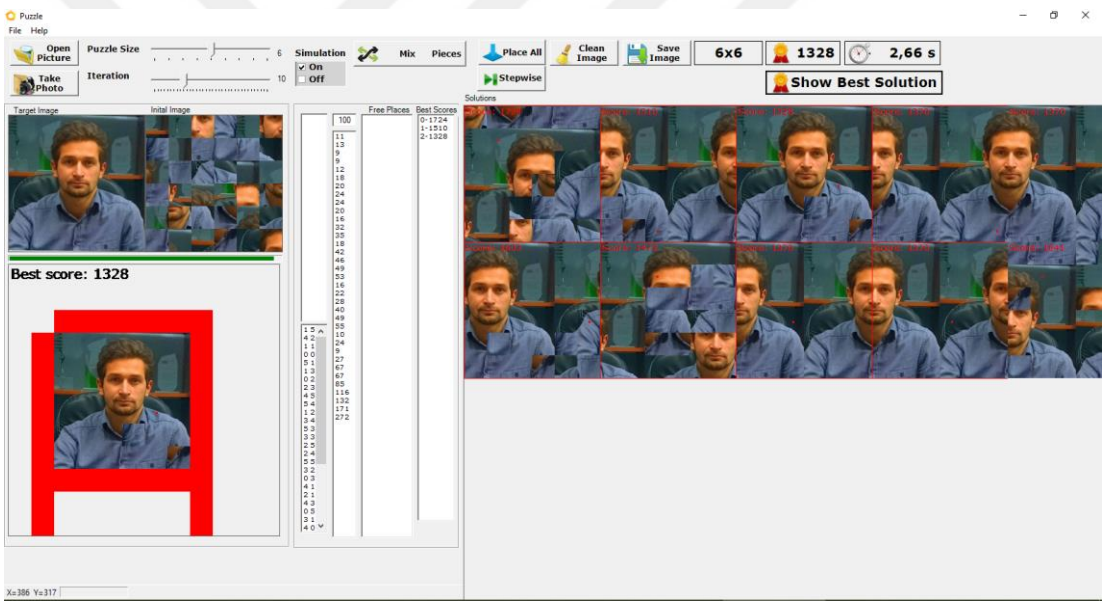

Figure 3.4 New GUI with 6x6 pieces puzzle solutions

Example 2:

In this example another webcam image is taken and sliced as 7x7 pieces as shown in Figure 3.5. This time only one solution attempt was made and get the correct answers. The final image is shown on the right side of the figure with possible future candidate adjacent cells shown with green colours, if we have more candidate pieces.

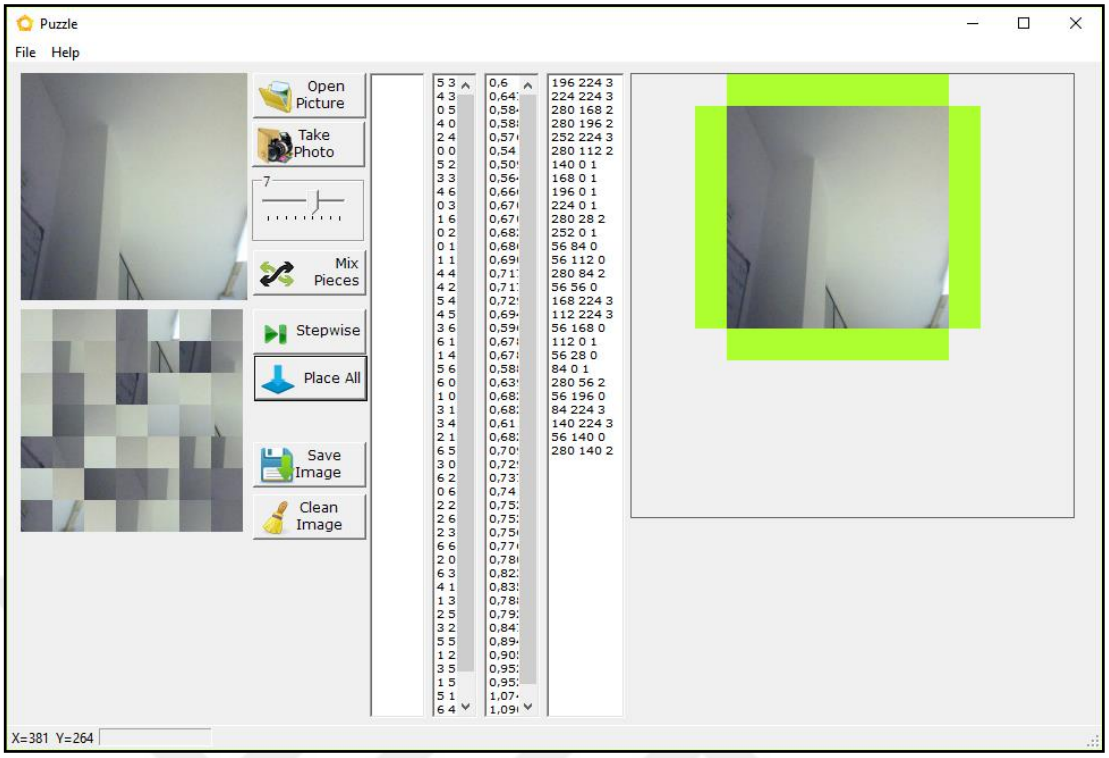

Figure 3.5 Successfully solved a 7x7 jigsaw puzzle example

Example 3:

In this example an image is uploaded and sliced as 8x8 pieces as shown in Figure 3.6. Then, implemantitons of group (Tetris) based jigsaw puzzle pieces are executed. In this example pieces are grouped and the number of pieces/groups is decreased to 16 from 64 without no errors. Also, the places of these successful match-ups can be changed, turned or replaced to finalize the solution.

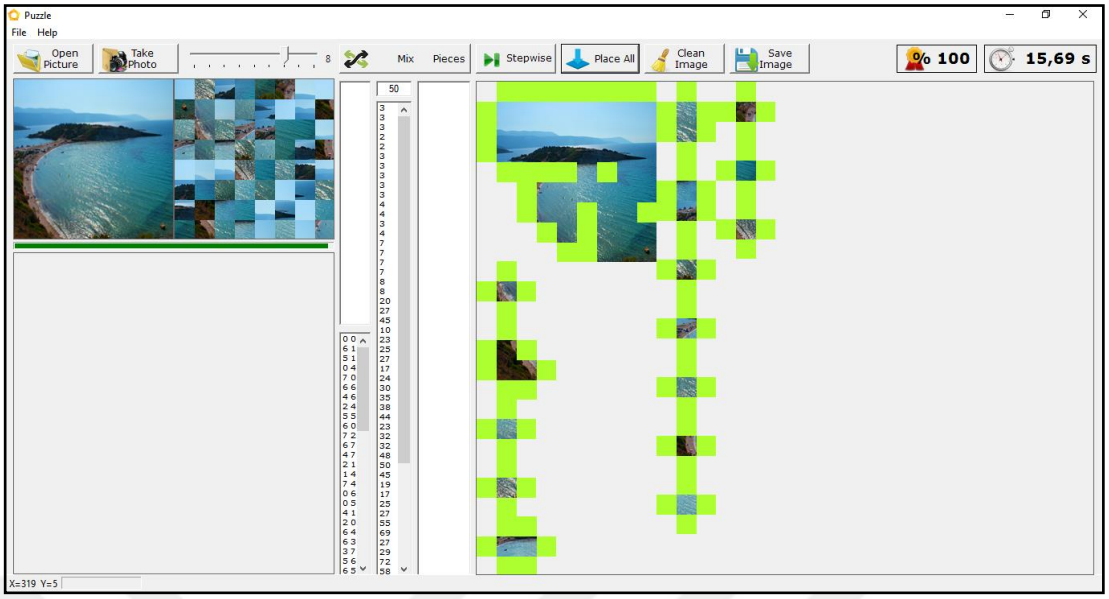

Figure 3.6. Group (Tetris game) based jigsaw puzzle pieces

3.4 Second Step for Solving the Jigsaw Puzzle

In this step, all the pieces of the jigsaw puzzle are checked for better matching scores with their neigbour pieces if the exact solutions haven't been obtained in the first step. Each pieces compatibility value with its neighbours is calculated. Since each piece has four sides or neighbors, it has four compatibilitiy value on each side. Sum of these four values will give the total compatibility value of that piece in that position. Pieces with higher compatibility values with their neighbours are grouped. If a piece has a very low compatibility values, becomes a single group. Then, single groups are swapped with each other to obtain better compatibility values. This operation will continue to get exact solution with highr compatibility or fitness value above a treshold. Pseudo code of the second step is given in Figure 3.7.

```
    while fitness<threshold_f do
      for j=0 to height do
          for i=0 to width do
            Calculate each pieces compatibility value: C(P0, Pcand, d)
          end for
      end for
      Group pieces with higher compatibility
      Perform crossover with groups and pieces with lower
compatibility value
    end while
```

Figure 3.7 Pseudo code of the proposed approach on second step

Compatibility values can be written on the pieces edges to simulate and understand the pieces compatibility. The pieces on the edges of the jigsaw has values with zero because they don't have any neighbour. Lower values means higher compatibility between the pieces edges as shown ing Figure 3.8. In this example edges which have compatibility values lower than 30 can be accepted as right or healty. In the same way edges which have compatibility values high than 30 accepted as wrong or unhealty. So these unhealty pieces and healthy pieces groups are swapped repeatedly to obtain more healty pieces. Finally we will obtain better results as shown on the right side in Figure 3.8.

Figure 3.8 Missing solution with unhealthy pieces (left) and exact solution with more healthy pieces and compatibility values

Another example presentation for 10x10 jigsaw puzzle is given in Figure 3.9. Input image or mixed pieces are given on the left side and approximate solution is given on the right side on the figure like grid.

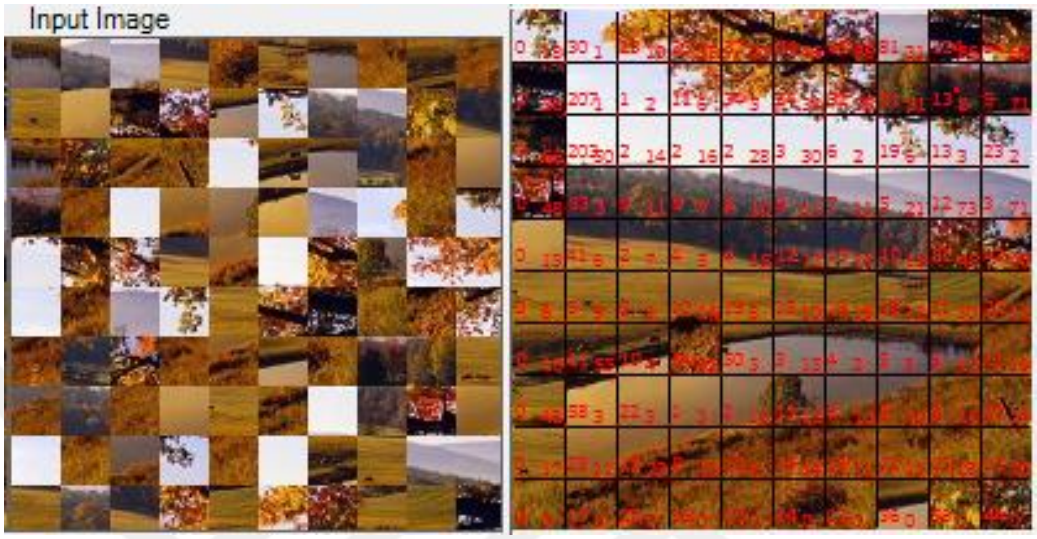

Figure 3.9 Input image or mixed pieces (left) approximate solution for 10x10 jigsaw puzzle with grid representation (right)

CHAPTER FOUR

RESULTS AND CONCLUSIONS

4.1 Results and Conclusions

Implementation and testing of jigsaw puzzle solving algorithm based on edge matching with RGB colours of edge pixels: decided to group the pieces first which have the maximum compatibility. Thus, it will prevent the possible mismatches and it will save time. After first grouping operation, some more grouping operations can be done till to obtain final single piece image. In each grouping operation step, compatibility threshold values can be adaptively reduced to obtain final image and, in each group, can be rotated and assembled according to their shapes.

Grouping or Tetris mode is also implemented for an alternative solution method. Step by step solving and simulations are carried out. The optimum solution of the first step is obtained from some initial solutions. Then, the compatibility scores of each piece and each edge are obtained and written on edges to detect the incompatibility of edges and pieces. These scores can be used for improving the optimum solution of the first step.

Beside this, each edge of the jigsaw pieces and jigsaw piece groups will be matchup and piece/group swapping algorithms with genetic algorithm can be implemented in the future. And finally, if there is missing jigsaw puzzle piece, the empty cell can be estimated by adjacent cells.

Each piece on the solution can be also checked again with their neighbour pieces. The most incompatible pieces places can be changed among themselves for better solutions.

REFERENCES

Billmeyer & Saltzman (2000). *Principles of color technology* (3rd ed.). USA: Wiley-Interscience.

Caroline (2019). *The 5 types of digital image files*, Retrieved January 5, 2019, from https://www.ivanexpert.com/blog/2010/05/the-5-types-of-digital-image-files-tiff-jpeg-gif-png-and-raw-image-files-and-when-to-use-each-one.

Davenport, A. J., Tsang, E. P.K. (1999). Solving constraint satisfaction sequencing problems by iterative repair. *In Proceedings of The Protocol Application of Constraint Technologies and Logic Programming*, 345–357.

Glassner, A. (2002*).* Putting the pieces together. *IEEE Computer Graphics and Applications*, (May/June), 76–86.

Goldberg, D., Malon, C., Bern, M. (2002). *A global approach to automatic solution of jigsaw puzzles*, SoCG'02, June 5-7, Barcelona, Spain.

Hatti, M. (2018). *Artificial intelligence in renewable energetic system*. USA: Springer.

Roy, S., Shawon, M.F.H. & Safa, Z.T. (2019). *A p-median closed loop model using tabu search for health care supply in Bangladesh,* Bachelor Thesis, Bangladesh University of Engineering and Technology, Bangladesh.

Tybon, R. (2004). *Generating solutions to the jigsaw puzzle problem*, Phd Thesis, Griffith University, Queensland, Australia.

Yao, F.H., Shao, G.F. (2003). A shape and image merging technique to solve jigsaw puzzles, *Pattern Recognition Letters 24*, 1819-1835.

WhatIs.com (2019), *RGB (red, green, and blue)*, Retrieved January 5, 2019, from https://whatis.techtarget.com/definition/RGB-red-green-and-blue.

Wikipedia.com (2019), Retrieved January 10, 2019, from https://www.wikipedia.org.

APPENDICES

Main Functions and Source Code of the Computer Program:

```
public int[] Benzerlik(Image Source,Image Target)
        {
            int[] array= new int[5];
//1:Sağında,2:Altında,3:Solunda,4:Üstünde
            int benzerlik= 0;
            Bitmap Sr= new Bitmap(Source);
            Bitmap Tr= new Bitmap(Target);

            for (int p= 1; p<= Sr.Height -1; p++) {if
(Math.Abs(Sr.GetPixel(Sr.Width -1,p).R -Tr.GetPixel(1,p).R)<10)
{benzerlik++;} }

            array[1]= benzerlik; benzerlik= 0;

            for (int p= 1; p<= Sr.Width -1; p++) {if
(Math.Abs(Sr.GetPixel(p,Sr.Height -1).R -Tr.GetPixel(p,1).R)<10)
{benzerlik++;} }

            array[2]= benzerlik; benzerlik= 0;

            for (int p= 1; p<= Sr.Height -1; p++) {if
(Math.Abs(Sr.GetPixel(1,p).R -Tr.GetPixel(Tr.Width -1,p).R)<10)
{benzerlik++;} }

            array[3]= benzerlik; benzerlik= 0;

            for (int p= 1; p<= Sr.Width -1; p++) {if
(Math.Abs(Sr.GetPixel(p,1).R -Tr.GetPixel(p,Tr.Height -1).R)<10)
{benzerlik++;} }

            array[4]= benzerlik; benzerlik= 0;
            return array;
        }
```

```
        private void YeniOyunClick(object sender,System.EventArgs e)
         {
             Parcala();
             Karistir();
             Temizle();
         }

         private void AboutClick(object sender,System.EventArgs e)
         {
             string tempX;
             tempX= "Studant Name    : Lütfi MUTLU" +
                  "\nStudent Number  : 2017900489" +
                  "\nSupervisor      : Prof. Dr. Efendi NASİBOĞLU";
             MessageBox.Show(temp,"M. Sc. Thesis");
         }

         private void buton16X_Click(object sender,EventArgs e)
         {
             int[] s= Benzerlik(buton1.Image,buton2.Image);
            // MessageBox.Show(yon +s[1].ToString() +" "
+s[2].ToString() +" " +s[3].ToString() +" "
+s[4].ToString(),"Sağında,Altında,Solunda,Üstünde");

             Rectangle rectangle1= new Rectangle(1,1,buton1.Width -
1,buton1.Height -1);
             Rectangle rectangle2= new Rectangle(mX,mY,buton1.Width -
1,buton1.Height -1);

CopyRegionIntoImage(buton1.Image,rectangle1,final.Image,rectangle2);
         }
         public static void CopyRegionIntoImage(Image
srcBitmap,Rectangle srcRegion,Image destBitmap,Rectangle hedefBolge)
         {
             using (Graphics grafikrD=
Graphics.FromImage(destBitmap))
             {
grD.DrawImage(srcBitmap,hedefBolge,srcRegion,GraphicsUnit.Pixel);
             }
         }
```

```
     private void buton17X_Click(object sender,EventArgs e)
        {
            buton1.Image=
(Bitmap)Bitmap.FromFile(Application.StartupPath +"\\Images\\1.jpg");
            buton2.Image=
(Bitmap)Bitmap.FromFile(Application.StartupPath +"\\Images\\2.jpg");
            buton3.Image=
(Bitmap)Bitmap.FromFile(Application.StartupPath +"\\Images\\3.jpg");
            buton4.Image=
(Bitmap)Bitmap.FromFile(Application.StartupPath +"\\Images\\4.jpg");
            buton5.Image=
(Bitmap)Bitmap.FromFile(Application.StartupPath +"\\Images\\5.jpg");
            buton6.Image=
(Bitmap)Bitmap.FromFile(Application.StartupPath +"\\Images\\6.jpg");
            buton7.Image=
(Bitmap)Bitmap.FromFile(Application.StartupPath +"\\Images\\7.jpg");
            buton8.Image=
(Bitmap)Bitmap.FromFile(Application.StartupPath +"\\Images\\8.jpg");
            final.Image=
(Bitmap)Bitmap.FromFile(Application.StartupPath
+"\\Images\\bos.jpg");
            butun.Image=
(Bitmap)Bitmap.FromFile(Application.StartupPath
+"\\Images\\bos.jpg");

            Rectangle rectangle1= new Rectangle(1,1,buton1.Width -
1,buton1.Height -1);
            //Rectangle rectangle2= new
Rectangle(200,200,buton1.Width -1,buton1.Height -1);

CopyRegionIntoImage(buton1.Image,rectangle1,butun.Image,rectangle1);
        }

        private void formMain_Load(object sender,EventArgs e)
        {
        butun.Image=(Bitmap)Bitmap.FromFile(Application.StartupPath
+"\\resimlerX\\dolu.jpg");
```

```
        yedek.Image=(Bitmap)Bitmap.FromFile(Application.StartupPath
+"\\resimlerX\\bos.jpg");

            trans= new Bitmap(final.Width,final.Height);
            Graphics grafikx= Graphics.FromImage(trans);
            gx.Clear(Color.FromArgb(0,0,0,0));
            final.Image= trans;

            trans= new Bitmap(yedek.Width,yedek.Height);
            gx= Graphics.FromImage(trans);
            gx.Clear(Color.FromArgb(0,0,0,0));
            yedek.Image= trans;

            Parcala();
            Karistir();
            Temizle();

            rekorPuzzle=new Bitmap(400,400);
            Graphics grafikg1= Graphics.FromImage(rekorPuzzle);
            gg1.Clear(Color.FromArgb(0,0,0,0));

            // this.CamCapture.CaptureHeight= this.butun.Height;
            // this.CamCapture.CaptureWidth= this.butun.Width;
            // CamCapture.Visible=false;

            modeUnrotated.Checked=true;
            modeRotated.Checked=false;
            modeTetris.Checked=false;

            checkedListBox1.SetItemChecked(0,true);
         //   Array.Clear(sol,0,15);

        }

        private void buton20_Click(object sender,EventArgs e)
        {
            int[] s= Benzerlik(buton3.Image,buton6.Image);
            string yon= "";
            int p= Array.IndexOf(s,s.Max());
```

```
            switch (p)
            {
                case 1: yon= "Sağında: "; break;
                case 2: yon= "Altında: "; break;
                case 3: yon= "Solunda: "; break;
                case 4: yon= "Üstünde: "; break;
            }

            MessageBox.Show(yon +s[1].ToString() +" "
+s[2].ToString() +" " +s[3].ToString() +" "
+s[4].ToString(),"Sağında,Altında,Solunda,Üstünde");

            Rectangle rectangle1= new Rectangle(1,1,buton1.Width -
1,buton1.Height -1);
            Rectangle rectangle2= new Rectangle(mX,mY,buton1.Width -
1,buton1.Height -1);

CopyRegionIntoImage(buton1.Image,rectangle1,final.Image,rectangle2);
        }

        public Bitmap birlestir(Bitmap[,] input)
        {
            int parcaX= input.GetLength(0);
            int parcaY= input.GetLength(1);
            Bitmap result= new Bitmap(gen*parcaX,yuk*parcaY);
            Rectangle rect0= new Rectangle(0,0,gen,yuk);

            for (int j= 0; j<parcaY; j++)
            {for (int i= 0; i<parcaX; i++)
             {
               Rectangle hedefBolge= new Rectangle(i * gen,j *
yuk,gen,yuk);
               using (Graphics grafikrD= Graphics.FromImage(result))
{grD.DrawImage(input[i,j],hedefBolge,rect0,GraphicsUnit.Pixel);}
              }
            }
            return result;
        }
```

```
        public Bitmap[,] Karistir(Bitmap[,] input)
        {
            int parcaX= input.GetLength(0);
            int parcaY= input.GetLength(1);
            Bitmap[,] result= new Bitmap[parcaX,parcaY];

            var a= new Random(DateTime.Now.Ticks.GetHashCode());

            int n, j, i= 0;
            while (n<Lcand.Items.Count) // Listeyi Karıştır.
            {
                int k= a.Next(n +1);
                string temp= Lcand.Items[k].ToString();
                Lcand.Items[k]= Lcand.Items[n];
                Lcand.Items[n]= temp;
                n++;
            }

            i= 0; j= 0;
            for (n= 0; n<Lcand.Items.Count; n++)
            {
                string[] xy= Lcand.Items[n].ToString().Split(null);
                result[i,j]=
input[int.Parse(xy[0]),int.Parse(xy[1])];

                i++;
                if (i>= parcaX) {j++; i= 0;}
            }

            return result;
        }

        public static void CopyIntoImage(Bitmap srcBitmap,Rectangle
srcRegion,ref Bitmap destBitmap,Rectangle hedefBolge)
        {
```

```
            using (Graphics grafikrD=
Graphics.FromImage(destBitmap))
            {
grD.DrawImage(srcBitmap,hedefBolge,srcRegion,GraphicsUnit.Pixel);
            }
        }

        public Bitmap[,] Parcala(Bitmap input,int parca)
        {
            Bitmap[,] result= new Bitmap[parca,parca];
            gen= input.Width / parca;
            yuk= input.Height / parca;
            progressBar.Maximum= parca * parca;

            for (int j= 0; j<parca; j++)
            {
                for (int i= 0; i<parca; i++)
                {
                    result[i,j]= Kes(input,i * gen,j * yuk,gen,yuk);
                }
            }

            Lcand.Items.Clear();

            for (int jj= 0; jj<parca; jj++) // Listeyi Doldur.
            {for (int ii= 0; ii<parca; ii++)
                {
                  Lcand.Items.Add(ii.ToString() +" "
+jj.ToString());
                }
             }

            mX= gen * parca; final.Width= 2 * mX;
            mY= yuk * parca; final.Height= 2 * mY;

            return result;
        }
```

```
        public Bitmap Kes(Bitmap input,int xKonum,int yKonum,int
width,int height)
        {
            Bitmap temp= (Bitmap)input;
            if (xKonum +width>input.Width) width= input.Width -
xKonum;
            if (yKonum +height>input.Height) height= input.Height -
yKonum;
            Rectangle rectX= new
Rectangle(xKonum,yKonum,width,height);
            input= (Bitmap)bmap.Clone(rectX,bmap.PixelFormat);
            return input;
        }

        public void OtoKes()
        {
            Rectangle rect0= new
Rectangle(0,0,gen*track.Value,yuk*track.Value);

            Bitmap input= (Bitmap)final.Image;
            Bitmap output= (Bitmap)yedek.Image;

            int xmin= 999,ymin= 999,x_maxx= 0,y_maxx= 0,width=
0,height= 0;
            for (int j= 1; j<final.Image.Height; j++)
            {
                for (int i= 1; i<final.Image.Width; i++)
                {
                    if (input.GetPixel(i,j) !=
Color.FromArgb(0,0,0,0))
                    {
                        if (i<xmin) {xmin= i;}
                        if (i>x_maxx) {x_maxx= i;}
                        if (j<ymin) {ymin= j;}
                        if (j>y_maxx) {y_maxx= j;}
                    }
                }
            }
```

```
            //minX++;
            maxX++;
           // minY++;
            maxY++;
            xmin= minX* mX / track.Value; ymin= minY * mX /
track.Value; x_maxx=maxX* mX / track.Value; y_maxx= maxY * mX /
track.Value;

            width= x_maxx -xmin;
            height= y_maxx -ymin; // input size
            Rectangle rectX = new Rectangle(xmin,ymin,width,height);

            PointF firstLocation= new PointF(xmin,ymin);
            PointF secondLocation= new PointF(posX -width,posY);
            Font arialFont= new Font("Verdana",10);
            using (Graphics grafikrD=
Graphics.FromImage(final.Image)) {grD.DrawString("Score: "
+sum.ToString(),arialFont,Brushes.Red,firstLocation);} //Yazı Ekle
            //using (Graphics grafikrD=
Graphics.FromImage(yedek.Image))
{grD.DrawString(sum.ToString(),arialFont,Brushes.Red,firstLocation);
}
//Yazı Ekle

            x_maxx= 0; y_maxx= 0;

// Büyük resim konumu için sıfırla
// ***** 1. Sütun ******

            for (int j= 1; j<yedek.Height; j++)
            {for (int i= 1; i<yedek.Width; i++)
                {if (output.GetPixel(i,j) !=
Color.FromArgb(0,0,0,0))
                    {if (i>x_maxx) {x_maxx= i;}
                     if (j>y_maxx) {y_maxx= j;}
                    }
                }
             }
```

```
            Rectangle hedefBolge= new
Rectangle(0,y_maxx,width,height);

            if (mode== "Tetris") {
            int n= 0;
            while (y_maxx +height>yedek.Height)
            {
                if (y_maxx +height>yedek.Height)

// ***** n. Sütun ******

                {
                    if (x_maxx +width +1<yedek.Width && sol[n]== 0)
{sol[n]= x_maxx +1;}
                    x_maxx= 0; y_maxx= 0;
                    for (int j= 1; j<yedek.Height; j++)
                    {
                        for (int i= sol[n]; i<yedek.Width; i++)
                        {
                            if (output.GetPixel(i,j) !=
Color.FromArgb(0,0,0,0))
                            {if (i>x_maxx) {x_maxx= i;} if
(j>y_maxx) {y_maxx= j;} }
                        }
                    }
                    hedefBolge= new
Rectangle(sol[n],y_maxx,width,height);
                }n++; if (n== 15) break;
            }
            }
            else
            {
                hedefBolge= new Rectangle(posX,posY,width,height);
                posX += width;
                if (posX>yedek.Width) {posX= 0; posY += height;}

            }
```

```
            using (Graphics grafikrD=
Graphics.FromImage(yedek.Image)) {grD.DrawImage(input,hedefBolge,
rectX ,GraphicsUnit.Pixel);} //yeşile boya
            yedek.Refresh();

            Bitmap tempX= (Bitmap)yedek.Image;
            try {
            for (int iix= posX-width; iix<posX; iix++)
{tempX.SetPixel(iix,posY,Color.Red);
tempX.SetPixel(iix,posY+height,Color.Red);}
            for (int iiy= posY; iiy<posY +height; iiy++) {
tempX.SetPixel(posX,iiy,Color.Red); tempX.SetPixel(posX-
width,iiy,Color.Red);}
            }
            catch {}

            yedek.Image= tempX;

            trans= new Bitmap(final.Width +1,final.Height +1); //
Final Resmi Temizle
            Graphics grafik= Graphics.FromImage(trans);
            final.Image= trans;
            YerlesimVar=false;
            LFree.Items.Clear();
            LFree1.Items.Clear();
            LFree2.Items.Clear();
            LFree3.Items.Clear();
        }

        private void trackBar1_Scroll(object sender,EventArgs e)
        {
            labelTrack.Text= track.Value.ToString();
            iter.Maximum= track.Value * track.Value;

            //openFile.ShowDialog();
            //butun.Image=
(Bitmap)Bitmap.FromFile(openFile.FileName);
            //parcali.Image=Parcala((Bitmap)butun.Image,4)[1,1];
            //Karistir(Parcala((Bitmap)butun.Image,4));
```

```
            // ImageMatrix1=
Karistir(Parcala((Bitmap)butun.Image,trackBar1.Value));
            // parcali.Image= birlestir(ImageMatrix1); //
karistir(ImageMatrix1);
        }

        private void btnKaristir_click(object sender,EventArgs e)
        {
            Temizle(); Parcala();  Karistir();
            mboyut= (int)Math.Round(Math.Sqrt(Lcand.Items.Count));
            btnBoyut.Text= mboyut.ToString() +"x"
+mboyut.ToString();

           // btnTemizle.PerformClick();
        }

        public void Parcala()
        {
            ImageMatrix1= Parcala((Bitmap)butun.Image,track.Value);
//Parçalı Input Image Matrix oluştur
            yesil= new Bitmap(mX / track.Value,mY / track.Value);
            Graphics grafik= Graphics.FromImage(yesil);
            g.Clear(Color.GreenYellow);
        }

        public void Karistir()
        {
            if (Lcand.Items.Count<1)
            {Parcala();}
            ImageMatrix1= Karistir(ImageMatrix1); // Karıştırılmış
Parçalı Yeni Input Image Matrix
            parcali.Image= birlestir(ImageMatrix1);
            Lplaced.Items.Clear(); listRekor.Items.Clear(); sum= 0;
            LFree.Items.Clear();YerlesimVar=false;
            LFree1.Items.Clear();
            LFree2.Items.Clear();
            LFree3.Items.Clear();
        }
```

```
        public void Temizle()
        {
            trans= new Bitmap(final.Width+1,final.Height+1);
            Graphics grafik= Graphics.FromImage(trans);
            final.Image= trans;

            kirmizi= new Bitmap(mX / track.Value,mY / track.Value);
            Graphics grafikg= Graphics.FromImage(kirmizi);
            gg.Clear(Color.Red);

            foreach (var item in Lplaced.Items)
{Lcand.Items.Add(item);}
            Lplaced.Items.Clear(); listRekor.Items.Clear(); sum= 0;
YerlesimVar=false;
            LFree.Items.Clear();
            LFree1.Items.Clear();
            LFree2.Items.Clear();
            LFree3.Items.Clear();
            mX= 200; mY= 200;
            Array.Clear(sol,0,sol.Length);

            matrix= new int[2 * track.Value,2 * track.Value];
            bosyer= 1;
            //minX= mboyut -1;
            // minY= mboyut -1;
            minX= 999; minY= 999; maxX= 0; maxY= 0;
            boyandiX=false; boyandiY=false;

        }

        private void buton25_Click(object sender,EventArgs e)
//Eskiler
        {
            parcali.Image= birlestir(ImageMatrix1);
            Lplaced.Items.Clear(); listRekor.Items.Clear(); sum= 0;
//              LFree.Items.Clear();
        }
```

```
        public void Karistirmadan(int it)
        {
            if (it== 0)
            {
                LcandTemp.Items.Clear();
                for (int i= 0; i<Lcand.Items.Count; i++)
{LcandTemp.Items.Add(Lcand.Items[i]);} // Yedekle
            }

            Temizle();
            Lcand.Items.Clear();
            for (int i= 0; i<LcandTemp.Items.Count; i++)
{Lcand.Items.Add(LcandTemp.Items[i]);}

            mX= gen * track.Value; final.Width= 2 * mX;
            mY= yuk * track.Value; final.Height= 2 * mY;
            ilkYerlestir(it);

            //Parcala();
            //Karistir();
            //mboyut= (int)Math.Round(Math.Sqrt(Lcand.Items.Count));
            //btnBoyut.Text= mboyut.ToString() +"x"
+mboyut.ToString();
        }

        private void TekBirlestir_Click(object sender,EventArgs e)
        {if(Lcand.Items.Count>0) Yerlestir(1);}
        private void HepsiniBirlestir_Click(object sender,EventArgs
e)
        {
            stopwatch.Restart();

            for (int it=0; it<iter.Value; it++)
          {
                Karistirmadan(it);
                //if (it != 0) btnKaristir.PerformClick();

            int limit= 1;
            if (mode=="Tetris") {limit= Lcand.Items.Count;}
```

```
//Parça sayısı
// Bütün parçaları yerleştir

           for (int ii=0; ii<limit; ii++)
            {
              if (Lcand.Items.Count>0)
             {
              Yerlestir(1);
              Yerlestir(Lcand.Items.Count);
             }
            }

            //buton16.Text= "% " +Math.Round((1000-sum /
listRekor.Items.Count)/10).ToString();
                buton16.Text= sum.ToString();
                if (sum<TopRekor)
            {
                TopRekor= sum; //Yeni Rekor
                Bitmap input= (Bitmap)final.Image;
                rekorPuzzle= (Bitmap)input.Clone();
                    PointF firstLocation= new PointF(0,0);
                    Font arialFont= new Font("Verdana",15);
                    using (Graphics grafikrD=
Graphics.FromImage(rekorPuzzle)) {grD.DrawString("Best score: "
+sum.ToString(),arialFont,Brushes.Black,firstLocation);}
//Yazı Ekle

//using (Graphics grafikrD= Graphics.FromImage(yedek.Image))
{grD.DrawString(sum.ToString(),arialFont,Brushes.Red,firstLocation);
}
//Yazı Ekle

                listBox1.Items.Add(it +"-" +sum);
//listBox1.Items.Add(listRekor.Items.Count.ToString());

               // yedek.Image= (Bitmap)input.Clone();

            }
```

```
                OtoKes();
                stopwatch.Stop();
            buton20.Text=
Math.Round(stopwatch.Elapsed.TotalSeconds,2).ToString() +" s";
            this.Refresh();
          }
        }

        public void Yerlestir(int adet) // x adet parça yerleştir
        {
            Bitmap[,] input= ImageMatrix1; // Input Image Matrix
            int xx0= 0,yy0= 0,x1=0,y1=0;

            string[] xyy0,xy1;
            if (YerlesimVar)
// Daha önce eklenmiş ise ListFinal'in son elemanı=XYY0

            {
                xyy0= Lplaced.Items[Lplaced.Items.Count -
1].ToString().Split(null);
                xx0= int.Parse(xyy0[0]);
                yy0= int.Parse(xyy0[1]);

                for (int n= 0; n<adet; n++)
// kaç adet yerleştirilecekse (1 adet veya hepsi)
                {
                    int rekor= 99999; int yn= 0,rekXX0= 0,rekYY0=
0,rekX1= 0,rekY1= 0;
                    //for (int nj= 0; nj<LFree1.Items.Count; nj++)
// Lfree: Bütün boş yerlere bak. LFree1.Items.Count
                    //{
                    //    //xyy0=
LFree.Items[nj].ToString().Split(null);
                    //    xx0= (int)LFree1.Items[nj]; yy0=
(int)LFree2.Items[nj]; int yon= (int)LFree3.Items[nj];

                    //    for (int ni= 0; ni<Lcand.Items.Count;
ni++)
// Lcand: Bütün adayları dene
```

```
                    //    {
                    //        xy1=
Lcand.Items[ni].ToString().Split(null); x1= int.Parse(xy1[0]); y1=
int.Parse(xy1[1]);
                    //        double puan=
YerlesimPuani(xx0,yy0,input[x1,y1],yon);
                    //        if (puan<rekor) {rekor= puan; rekXX0=
xx0; rekYY0= yy0; rekX1= x1; rekY1= y1; yn= yon;}//
Array.IndexOf(s,s.Min());}
                    //    }
                    //}
                    for (int y= 0; y<2 * mboyut-1; y++)
                    {
                        for (int x= 0; x<2 * mboyut-1; x++)
                        {
                            if (matrix[x,y]== 2)
// 1: Dolu,2: Boş/Aday ise
                            {
                                for (int ni= 0;
ni<Lcand.Items.Count; ni++)
// Lcand: Bütün adayları dene
                                {
                                    xy1=
Lcand.Items[ni].ToString().Split(null);
                                    x1= int.Parse(xy1[0]);
//x1: x ekseni parça no
                                    y1= int.Parse(xy1[1]);
//y1: y ekseni parça no

                                    //int puan=
YerlesimPuani(x,y,input[x1,y1]);
//input[]: Input Image Matrix[] içinden parça x1,y1

                                    int[]puanlar=
Kontrol1(input[x1,y1],x,y);
                                    int puan= puanlar[0] +puanlar[1]
+puanlar[2] +puanlar[3];
                                    int result= puanlar.Count(i=>i
!= 0);
```

```
                                    puan= puan/ result;
                                    Console.WriteLine("Puan= "
+puan);
                                    if (puan<rekor) {rekor= puan;
rekXX0= x; rekYY0= y; rekX1= x1; rekY1= y1;} //
Array.IndexOf(s,s.Min());}
                               }
                                }
                           }
                      }

                      progressBar.Value= progressBar.Maximum -
Lcand.Items.Count;
                      //Console.WriteLine("Rekor(" +rekX1 +"-" +rekY1
+")=" +rekor);
                      listRekor.Items.Add(rekor);
                      sum += rekor;
                      buton16.Text= sum.ToString();

                 if (mode !="Tetris" ||
rekor<Convert.ToDouble(limit.Text))
                      {

Yapistir(rekXX0,rekYY0,yn,input[rekX1,rekY1]);
// Sim. aktif ise parça yerleştir

      if (Lcand.Items.Contains(rekX1 +" " +rekY1))
{Lcand.Items.Remove(rekX1 +" " +rekY1); Lplaced.Items.Add(rekX1 +" "
+rekY1);}
                      }
                      else
                      {
                          OtoKes();
                          break;
                      }
                 }
```

```
                if (mode== "Tetris" && Lcand.Items.Count== 0)
{
OtoKes();
}
            }
            else // ilk ekleme ve boyama
            {
                if (mode== "Tetris" && Lcand.Items.Count== 0)
                {
                    OtoKes(); //?
                }
                else //ilk boyama
                {
                    ilkYerlestir(0);
                }
            }

            for (int ax=0; a<2*mboyut; ax++)
            {
             for (int bx=0; b<2*mboyut; bx++)
                {
                  Console.Write(matrix[bx,ax] +" ");
                }
                Console.WriteLine();
            }
        }

// ilk veya iterasonda siradaki
        public void ilkYerlestir(int ilk)
        {

            if (Lcand.Items.Count<1) btnKaristir.PerformClick();

            Bitmap[,] input= ImageMatrix1;
            int xx0= 0,yy0= 0,x1= 0,y1= 0;

            string[] xyy0,xy1;
```

```
            xyy0= Lcand.Items[ilk].ToString().Split(null); //
Listedeki ilk elemanı yerleştir.
            xx0= int.Parse(xyy0[0]);
            yy0= int.Parse(xyy0[1]);
            Lplaced.Items.Add(Lcand.Items[ilk]);
            Lcand.Items.RemoveAt(ilk);

           if (checkedListBox1.GetItemChecked(0))
            {
             Rectangle rect0= new Rectangle(0,0,gen,yuk);
             Rectangle hedefBolge= new Rectangle(mX,mY,gen,yuk);
             using (Graphics grafikrD=
Graphics.FromImage(final.Image))
{grD.DrawImage(input[xx0,yy0],hedefBolge,rect0,GraphicsUnit.Pixel);}

             rect0= new Rectangle(0,0,gen/10,yuk/10);
             hedefBolge= new Rectangle(mX+gen / 2,mY+yuk /
2,gen/10,yuk/10);
             using (Graphics grafikrD=
Graphics.FromImage(final.Image))
{grD.DrawImage(kirmizi,hedefBolge,rect0,GraphicsUnit.Pixel);}
             final.Refresh();
            }
            Cercevele(mboyut,mboyut);

            Ekle((mX -gen) +" " +mY +" " +0); //sağı dolu
            Ekle(mX +" " +(mY -yuk) +" " +1); //altı dolu
            Ekle((mX +gen) +" " +mY +" " +2); //solu dolu
            Ekle(mX +" " +(mY +yuk) +" " +3); //üstü dolu

            YerlesimVar=true;
        }
        public void Yapistir(int rekXX0,int rekYY0,int yn,Bitmap
input) // boyama ve liste güncelleme
        {
            if (checkedListBox1.GetItemChecked(0)) // Simulasyon
aktif ise parçayı yerleştir/yapıştır.
            {
                Rectangle rect0= new Rectangle(0,0,gen,yuk);
```

```
                Rectangle hedefBolge= new Rectangle(rekXX0 *
gen,rekYY0 * yuk,gen,yuk);
                using (Graphics grafikrD=
Graphics.FromImage(final.Image))
{grD.DrawImage(input,hedefBolge,rect0,GraphicsUnit.Pixel);}
            }

            Cercevele(rekXX0,rekYY0);
            if (checkedListBox1.GetItemChecked(0)) final.Refresh();

            //string yon= "";
            //switch (yn) //temas eden kenarı sil ve yeni çevresini
ekle
            //{
            //    case 0:
            //        if (LFree.Items.Contains((rekXX0) +" " +rekYY0
+" " +yn)) {LFree.Items.Remove((rekXX0) +" " +rekYY0 +" " +yn);} //
kendini sil +artık sağa bakma
            //        if (rekXX0 -gen +10>0 &&
!LFree.Items.Contains((rekXX0 -gen) +" " +rekYY0 +" " +0))
{Ekle((rekXX0 -gen) +" " +rekYY0 +" " +0);}//sonra sağa bak
            //        if (rekYY0 -yuk +10>0 &&
!LFree.Items.Contains(rekXX0 +" " +(rekYY0 -yuk +" " +1)))
{Ekle(rekXX0 +" " +(rekYY0 -yuk) +" " +1);}
            //        if (rekYY0 +yuk +10<final.Height &&
!LFree.Items.Contains(rekXX0 +" " +(rekYY0 +yuk) +" " +3))
{Ekle(rekXX0 +" " +(rekYY0 +yuk) +" " +3);}
            //        yon= "Eskisi Sağda Yenisi Solda ";
            //        break;
            //    case 1:
            //        if (LFree.Items.Contains(rekXX0 +" " +rekYY0
+" " +yn)) {LFree.Items.Remove(rekXX0 +" " +rekYY0 +" " +yn);} //
kendini sil +artık alta bakma
            //        if (rekXX0 -gen +10>0 &&
!LFree.Items.Contains((rekXX0 -gen) +" " +rekYY0 +" " +0))
{Ekle((rekXX0 -gen) +" " +rekYY0 +" " +0);} //sola kay,sağına bak
            //        if (rekXX0 +gen +10<final.Width &&
!LFree.Items.Contains((rekXX0 +gen) +" " +rekYY0 +" " +2))
{Ekle((rekXX0 +gen) +" " +rekYY0 +" " +2);}//sağa kay,soluna bak
```

```
            //        if (rekYY0 -yuk +10>0 &&
!LFree.Items.Contains(rekXX0 +" " +(rekYY0 -yuk) +" " +yn))
{Ekle(rekXX0 +" " +(rekYY0 -yuk) +" " +yn);}
            //        yon= "Eskisi Altta Yenisi Üstte ";
            //        break;
            //    case 2:
            //        if (LFree.Items.Contains(rekXX0 +" " +rekYY0
+" "+yn)) {LFree.Items.Remove(rekXX0 +" " +rekYY0 +" " +yn);} //
kendini sil +artık sola bakma
            //        if (rekXX0 +gen +10<final.Width &&
!LFree.Items.Contains((rekXX0 +gen) +" " +rekYY0 +" " +yn))
{Ekle((rekXX0 +gen) +" " +rekYY0 +" " +yn);}
            //        if (rekYY0 -yuk +10>0 &&
!LFree.Items.Contains(rekXX0 +" " +(rekYY0 -yuk +" " +1)))
{Ekle(rekXX0 +" " +(rekYY0 -yuk) +" " +1);}
            //        if (rekYY0 +yuk +10<final.Height &&
!LFree.Items.Contains(rekXX0 +" " +(rekYY0 +yuk) +" " +3))
{Ekle(rekXX0 +" " +(rekYY0 +yuk) +" " +3);}
            //        yon= "Eskisi Solda Yenisi Sağda ";
            //        break;
            //    case 3:
            //        if (LFree.Items.Contains(rekXX0 +" " +rekYY0
+" " +yn)) {LFree.Items.Remove(rekXX0 +" " +rekYY0 +" " +yn);} //
kendini sil +üste bakma
            //        if (rekXX0 -gen +10>0 &&
!LFree.Items.Contains((rekXX0 -gen) +" " +rekYY0 +" " +0))
{Ekle((rekXX0 -gen) +" " +rekYY0 +" " +0);} //sola kay,sağına bak
            //        if (rekXX0 +gen +10<final.Width &&
!LFree.Items.Contains((rekXX0 +gen) +" " +rekYY0 +" " +2))
{Ekle((rekXX0 +gen) +" " +rekYY0 +" " +2);}//sağa kay,soluna bak
            //        if (rekXX0 +gen +10<final.Width && rekYY0
+yuk<final.Height && !LFree.Items.Contains(rekXX0 +" " +(rekYY0
+yuk) +" " +yn)) {Ekle(rekXX0 +" " +(rekYY0 +yuk) +" " +yn);}
            //        yon= "Eskisi Üstte Yenisi Altta ";
            //        break;
            //}
        }
```

```
        public void Cercevele(int xx,int yy)
        {
         //if (matrix[xx,yy]== 2) {SilYeni(xx,yy);}
         matrix[xx,yy]= 1; // 1: Dolu,2: Aday
            if (xx<minX) minX= xx;
            if (yy<minY) minY= yy;
            if (xx>maxX) maxX= xx;
            if (yy>maxY) maxY= yy;
            bosyer--;

            for (int y=0; y<2* mboyut; y++)
           {for (int x=0; x<2*mboyut; x++)
             {
               if (matrix[x,y]== 1)  // 1: Dolu,2: Aday
                    {
                        try
                        {
                          if (matrix[x+1,y]== 0)
{ EkleYeni(x+1,y,2); } // Sağda
                          if (matrix[x-1,y]== 0)
{ EkleYeni(x-1,y,0); } // Solda
                          if (matrix[x,y+1]== 0)
{ EkleYeni(x,y+1,3); } // Üstte
                          if (matrix[x,y-1]== 0)
{ EkleYeni(x,y-1,1); } // Altta
                        }
                    }
             }
           }
        }

        public void SilYeni(int x, int y)
        {



        }
        public void EkleYeni(int x,int y,int z)
//z: 0 Boş,1,Dolu,2 Aday
        {
```

```
          //bool sinirda= KontrolEt(x,y,z);
            if (icerdeMi(x,y,z) &&  matrix[x,y]== 0)
           {
            bosyer++;
            matrix[x,y]= 2;
// 0:Boş,1: Dolu,2: Aday,3: Duvar

            int xx0= x * gen;
            int yy0= y * yuk;

                if (checkedListBox1.GetItemChecked(0))
                {
                  Bitmap temp= (Bitmap)final.Image;
                  Color renk= temp.GetPixel(xx0,yy0);
                  Rectangle rect0= new
Rectangle(0,0,yesil.Width,yesil.Height);
                  Rectangle hedefBolge= new
Rectangle(xx0,yy0,yesil.Width,yesil.Height);
                  using (Graphics grafikrD=
Graphics.FromImage(final.Image))
{grD.DrawImage(yesil,hedefBolge,rect0,GraphicsUnit.Pixel);} //yeşile
boya
                  final.Refresh();
                }

                String message= xx0 +" " +yy0 +" " +z;
            LFree.Items.Add(message);
            LFree1.Items.Add(xx0);
            LFree2.Items.Add(yy0);
            LFree3.Items.Add(z);

                //if (renk== Color.FromArgb(0,0,0,0))
//Boyanmadıysa/Saydamsa Boya
                //{
                ////LFree.Items.Add(message);
                //}
            }

        }
```

```
        public bool icerdeMi(int x,int y,int z)
//0 Boş,1,Dolu,2 Aday

        {
            bool sonuc=true;
            //int sayY= 0,sayX= 0;
            //if (sayX>= mboyut)
            //{
            //    sonuc=false;
            //    for (int i= minY; i<minY +mboyut +1; i++)
{Boya(x,i);}
            //    for (int i= minY; i<minY +mboyut +1; i++) {Boya(x
+mboyut +1,i);}
            //    for (int i= minY; i<minY +mboyut +1; i++) {Boya(x
-mboyut -1,i);}
            //}


            if (mode != "Tetris")
            {
                if (maxX -minX>= mboyut -1) {for (int i= minY;
i<minY +mboyut +1; i++) {boyandiX=true; Boya(minX -1,i); Boya(maxX
+1,i);} }
                if (maxY -minY>= mboyut -1) {for (int i= minX;
i<minX +mboyut +1; i++) {boyandiY=true; Boya(i,minY -1); Boya(i,maxY
+1);} }
            }
            return sonuc;
        }
        public void Boya(int x,int y)
        {
         matrix[x,y]= 3;

         if (checkedListBox1.GetItemChecked(0))
          {
            int xx0= x * gen;
            int yy0= y * yuk;
            Bitmap temp= (Bitmap)final.Image;
            Color renk= temp.GetPixel(xx0,yy0);
```

```
            Rectangle rect0= new Rectangle(0
      0,kirmizi.Width,kirmizi.Height);
            Rectangle hedefBolge= new
Rectangle(xx0,yy0,kirmizi.Width,kirmizi.Height);
            using (Graphics grafikrD=
Graphics.FromImage(final.Image))
{grD.DrawImage(kirmizi,hedefBolge,rect0,GraphicsUnit.Pixel);}
//yeşile boya
            final.Refresh();

            //Console.Write(" minX="); Console.Write(minX);
            //Console.Write(" minY="); Console.Write(minY);
            //Console.Write(" maxX="); Console.Write(maxX);
            //Console.Write(" maxY="); Console.Write(maxY);

          }
        }
        public void Ekle(String message)
        {
            //String[] xyy0= message.Split(null);
            //Bitmap temp= (Bitmap)final.Image;
            //Color renk= temp.GetPixel(xx0,yy0);

            //if (renk== Color.FromArgb(0,0,0,0))
//Boyanmadıysa/Saydamsa Boya
            //{
                //LFree.Items.Add(message);
                //using (Graphics grafikrD=
Graphics.FromImage(final.Image))
{grD.DrawImage(yesil,hedefBolge,rect0,GraphicsUnit.Pixel);} //yeşile
boya
                //final.Refresh();
            //}
        }
        public void Sil(String message)
        {
            String[] xyy0= message.Split(null);
            int xx0=int.Parse(xyy0[0]);
```

```
            int yy0=int.Parse(xyy0[1]);
            int yon=int.Parse(xyy0[2]);
            Bitmap temp= (Bitmap)final.Image;
            Color renk= temp.GetPixel(xx0,yy0);

            if (renk== Color.GreenYellow) //Boyandıysa Sil
            {
                LFree.Items.Add(message);
                Rectangle rect0= new
Rectangle(0,0,yesil.Width,yesil.Height);
          Rectangle hedefBolge= new
Rectangle(xx0,yy0,yesil.Width,yesil.Height);
                using (Graphics grafikrD=
Graphics.FromImage(final.Image))
{grD.DrawImage(yesil,hedefBolge,rect0,GraphicsUnit.Pixel);} //yeşile
boya
                final.Refresh();
            }
        }
        public int YerlesimPuani(int xa,int ya,Image input)
        {
            // int[] array= new int[4];
//0:Sağında,1:Altında,2:Solunda,3:Üstünde
            double valueR= 0,valueG= 0,valueB= 0;
            Bitmap Sr= new Bitmap(input); //Input Image Matrix[]
içinden parça x1,y1
            Bitmap Tr= Kes((Bitmap)final.Image,0,0,gen,yuk);

            if (matrix[xa +1,ya]== 1) // Sağı Dolu
            {
                Tr= Kes((Bitmap)final.Image,xa*gen +gen,ya *
yuk,gen,yuk);
                for (int p= 1; p<= Tr.Height -1; p++)
                {
                    valueR += Math.Abs(Sr.GetPixel(Sr.Width -1,p).R
-Tr.GetPixel(1,p).R);
                    valueG += Math.Abs(Sr.GetPixel(Sr.Width -1,p).G
-Tr.GetPixel(1,p).G);
```

```
                    valueB += Math.Abs(Sr.GetPixel(Sr.Width -1,p).B
-Tr.GetPixel(1,p).B);
                }
            }
            else if (xa>1 &&  matrix[xa -1,ya]== 1) // Solda
            {
                Tr= Kes((Bitmap)final.Image,xa * gen -gen,ya *
yuk,gen,yuk);
                for (int p= 1; p<= Tr.Height -1; p++)
                {
                    valueR += Math.Abs(Sr.GetPixel(1,p).R -
Tr.GetPixel(Tr.Width -1,p).R);
                    valueG += Math.Abs(Sr.GetPixel(1,p).G -
Tr.GetPixel(Tr.Width -1,p).G);
                    valueB += Math.Abs(Sr.GetPixel(1,p).B -
Tr.GetPixel(Tr.Width -1,p).B);
                }
            }
            else if (ya>1 && matrix[xa,ya -1]== 1) // Üstte
            {
                Tr= Kes((Bitmap)final.Image,xa * gen,ya * yuk -
yuk,gen,yuk);
                for (int p= 1; p<= Tr.Width -1; p++)
                {
                    valueR += Math.Abs(Sr.GetPixel(p,1).R -
Tr.GetPixel(p,Tr.Height -1).R);
                    valueG += Math.Abs(Sr.GetPixel(p,1).G -
Tr.GetPixel(p,Tr.Height -1).G);
                    valueB += Math.Abs(Sr.GetPixel(p,1).B -
Tr.GetPixel(p,Tr.Height -1).B);
                }
            }
            else if (matrix[xa,ya +1]== 1) // Altta
            {
                Tr= Kes((Bitmap)final.Image,xa * gen,ya * yuk
+yuk,gen,yuk);
                for (int p= 1; p<= Tr.Width -1; p++)
                {
```

```
                    valueR += Math.Abs(Sr.GetPixel(p,Sr.Height -1).R
-Tr.GetPixel(p,1).R);
                    valueG += Math.Abs(Sr.GetPixel(p,Sr.Height -1).G
-Tr.GetPixel(p,1).G);
                    valueB += Math.Abs(Sr.GetPixel(p,Sr.Height -1).B
-Tr.GetPixel(p,1).B);
                }
            }

            valueR= valueR / 255;
            valueG= valueG / 255;
            valueB= valueB / 255;
            int sonuc= Convert.ToInt32(1000* (valueR +valueG
+valueB) / (3*gen));

            return sonuc;
        }

        private void buton21_Click(object sender,EventArgs e)
        {
            final.Image= rekorPuzzle;
            //buton16.Text= TopRekor.ToString();
            final.Refresh();
            this.Refresh();
        }

        private void checkedListBox1_Click(object sender,EventArgs
e)       {
checkedListBox1.SetItemChecked(0,!checkedListBox1.GetItemChecked(0))
;
checkedListBox1.SetItemChecked(1,!checkedListBox1.GetItemChecked(1))
;
        }

        private void buton22_Click(object sender,EventArgs e)
        {
            Bitmap tempRek;//= rekorPuzzle;
            tempRek= new Bitmap(400,400);
            Graphics grafikg1= Graphics.FromImage(tempRek);
```

```
            //final.Image= rekorPuzzle;
            //buton16.Text= TopRekor.ToString();
            //final.Refresh();
            //this.Refresh();

            Rectangle rect0= new Rectangle(0,0,gen * track.Value,yuk
* track.Value);
            Bitmap input= (Bitmap)rekorPuzzle;


            int xmin= 999,ymin= 999,x_maxx= 0,y_maxx= 0,width=
0,height= 0;
            for (int jx = 1; jx<rekorPuzzle.Height; jx++)
            {
                for (int ix = 1; ix<rekorPuzzle.Width; ix++)
                {
                if (input.GetPixel(ix,j)!= Color.FromArgb(0,0,0,0))
                    {
                        if (ix < xmin) {xmin = ix;}
                        if (ix > x_maxx) {x_maxx = ix;}
                        if (jx < ymin) {ymin = jx;}
                        if (jx > y_maxx) {y_maxx = jx;}
                    }
                }
            }
            //minX++;
            maxX++;
            // minY++;
            maxY++;
            xmin= minX * mX / track.Value; ymin= minY * mX /
track.Value; x_maxx= maxX * mX / track.Value; y_maxx= maxY * mX /
track.Value;

            width= x_maxx -xmin;
            height= y_maxx -ymin; // input size
            Rectangle rectX = new Rectangle(xmin,ymin,width,height);
            Rectangle hedefBolge= new Rectangle(0,0,width,height);
```

```
            using (Graphics grafikrD= Graphics.FromImage(tempRek))
{grD.DrawImage(input,hedefBolge, rectX ,GraphicsUnit.Pixel);}
//yeşile boya
            final.Refresh();

            //**********************
            //tempRek= (Bitmap)final.Image;
            //tempRek= (Bitmap)input.Clone();

            for (int y= 2; y<(mboyut +1) -2; y++)
            {
                for (int xx= 2; xx<(mboyut * gen) -2; xx++)
                {
                    int value= getValX(xx,y * yuk);
                    if (value>100) tempRek.SetPixel(xx,y *
yuk,Color.FromArgb(255,255,0,0));
                    else tempRek.SetPixel(xx,y *
yuk,Color.FromArgb(255,255,255,255));

                    //rekorPuzzle.GetPixel(x,y)
                }
            }


            for (int x= 2; x<(mboyut +1) -2; x++)
            {
                for (int yy= 2; yy<(mboyut * yuk) -2; yy++)
                {
                    int value= getValY(x*gen,yy);
                    if (value>100)
tempRek.SetPixel(x * gen,yy,Color.FromArgb(255,255,0,0));
                    else
tempRek.SetPixel(x * gen,yy,Color.FromArgb(255,255,255,255));
                }
            }

            final.Image= tempRek;
            buton16.Text= TopRekor.ToString();
            final.Refresh();
```

```
            this.Refresh();
        }
        public int getValY(int x,int y)
        {
            int value=0;
            for (int b= 1; b<= gen; b++)
            {
                value += Math.Abs(trans.GetPixel(x +b,y -0).R -
trans.GetPixel(x +b,y +1).R);
                //value += Math.Abs(trans.GetPixel(x +b,y -0).G -
trans.GetPixel(x +b,y +1).G);
                //value += Math.Abs(trans.GetPixel(x +b,y -0).B -
trans.GetPixel(x +b,y +1).B);
            }
            value= value/yuk;
            //if (value>55) value= 255;
            //else value= 0;
            return value;
        }
        public int getValX(int x,int y)
        {
            int value=0;
            for (int a=1; a<=yuk; a++)
            {
            value += Math.Abs(trans.GetPixel(x -0,y +a).R -
trans.GetPixel(x +1,y +a).R);
            //value += Math.Abs(trans.GetPixel(x -0,y +a).G -
trans.GetPixel(x +1,y +a).G);
           // value += Math.Abs(trans.GetPixel(x -0,y +a).B -
trans.GetPixel(x +1,y +a).B);
            }

            value= value/yuk ;
            //if (value>88) value= 255;
            //else value= 0;
            return value;
        }
```

```
        private void buton23_Click(object sender,EventArgs e)
        {
            Bitmap tempRek= new Bitmap(width,height);
            Graphics grafik= Graphics.FromImage(tempRek);
            final.Image= tempRek;

            //Bitmap tempRek= (Bitmap)final.Image;
            //tempRek= trans;
            //final.Image= tempRek;
            //final.Refresh();
            //this.Refresh();
            tempRek= trans;

            for (int j= 0; j<width; j+=yuk)
            {
                for (int i=0; i<height; i+=gen)
                {

                    int val= getValY(i,j);
                    for (int JJ= 1; JJ<yuk; JJ++)
                    {
tempRek.SetPixel(i,j +JJ,Color.FromArgb(val,val,255 -val,255 -val));
                    }

                     //val= getValX(i,j);
                     //   for (int ii= 1; ii<gen; ii++)
                     //   {
                     //       tempRek.SetPixel(i
+ii,j,Color.FromArgb(val,val,255 -val,255 -val));
                     //   }


                }
            }

            final.Image= tempRek;
            final.Refresh();
            this.Refresh();
```

```
            //final.Image= input;
            //buton16.Text= TopRekor.ToString();
            ///final.Refresh();
        }

        private void buton25_Click_1(object sender,EventArgs e)
        {
            for (int y= 0; y<2 * mboyut -1; y++)
            {
                for (int x= 0; x<2 * mboyut -1; x++)
                {
                    if (matrix[x,y]== 1)
// 1: Doluların çevresini kontrol
                    {
                     int[] puanlar= YerlesimPuani4(x,y); //input[]:
Input Image Matrix[] içinden parça x1,y1
                     Console.WriteLine();
                     Console.Write(" Sağ= " +puanlar[0]);
                     Console.Write(" Alt= " +puanlar[1]);
                     Console.Write(" Sol= " +puanlar[2]);
                     Console.Write(" Ust= " +puanlar[3]);
                    }
                }
            }
        }

        public int[] YerlesimPuani4(int xa,int ya)
        {
            // int[] array= new int[4];
//0:Sağında,1:Altında,2:Solunda,3:Üstünde

            double valueR= 0,valueG= 0,valueB= 0;
            //Bitmap Sr= new Bitmap(input);
//Input Image Matrix[] içinden parça x1,y1
            Bitmap Sr= Kes((Bitmap)final.Image,xa * gen,ya *
yuk,gen,yuk);
            Bitmap Tr= Kes((Bitmap)final.Image,0,0,gen,yuk);
```

```
            int[] puanlar= {0,0,0,0 };// sağ-alt-sol-üst

            if (xa<mboyut-1) //sağ puan
            {
                Tr= Kes((Bitmap)final.Image,xa * gen +gen,ya *
yuk,gen,yuk);
                for (int p= 1; p<= Tr.Height -1; p++)
                {
                    valueR += Math.Abs(Sr.GetPixel(Sr.Width -1,p).R
-Tr.GetPixel(1,p).R);
                    valueG += Math.Abs(Sr.GetPixel(Sr.Width -1,p).G
-Tr.GetPixel(1,p).G);
                    valueB += Math.Abs(Sr.GetPixel(Sr.Width -1,p).B
-Tr.GetPixel(1,p).B);
                }
                valueR=1+valueR / 255; valueG= valueG / 255;
valueB= valueB / 255;
                puanlar[0]= Convert.ToInt32(1000 * (valueR +valueG
+valueB) / (3 * gen));
            }
            if (xa>0)
//sol puan
            {
                Tr= Kes((Bitmap)final.Image,xa * gen -gen,ya *
yuk,gen,yuk);
                for (int p= 1; p<= Tr.Height -1; p++)
                {
                    valueR += Math.Abs(Sr.GetPixel(1,p).R -
Tr.GetPixel(Tr.Width -1,p).R);
                    valueG += Math.Abs(Sr.GetPixel(1,p).G -
Tr.GetPixel(Tr.Width -1,p).G);
                    valueB += Math.Abs(Sr.GetPixel(1,p).B -
Tr.GetPixel(Tr.Width -1,p).B);
                }
                valueR= valueR / 255; valueG= valueG / 255; valueB=
valueB / 255;
                puanlar[2]= Convert.ToInt32(1000 * (valueR +valueG
+valueB) / (3 * gen));
            }
```

```
            if (ya<mboyut-1) //alt
            {
                Tr= Kes((Bitmap)final.Image,xa * gen,ya * yuk
+yuk,gen,yuk);
                for (int p= 1; p<= Tr.Width -1; p++)
                {
                    valueR += Math.Abs(Sr.GetPixel(p,Sr.Height -1).R
-Tr.GetPixel(p,1).R);
                    valueG += Math.Abs(Sr.GetPixel(p,Sr.Height -1).G
-Tr.GetPixel(p,1).G);
                    valueB += Math.Abs(Sr.GetPixel(p,Sr.Height -1).B
-Tr.GetPixel(p,1).B);
                }
                valueR= valueR / 255; valueG= valueG / 255;
valueB= valueB / 255;
                puanlar[1]= Convert.ToInt32(1000 * (valueR +valueG
+valueB) / (3 * gen));
            }

            if (ya>0) //ust
            {
                Tr= Kes((Bitmap)final.Image,xa * gen,ya * yuk -
yuk,gen,yuk);
                for (int p= 1; p<= Tr.Width -1; p++)
                {
                    valueR += Math.Abs(Sr.GetPixel(p,1).R -
Tr.GetPixel(p,Tr.Height -1).R);
                    valueG += Math.Abs(Sr.GetPixel(p,1).G -
Tr.GetPixel(p,Tr.Height -1).G);
                    valueB += Math.Abs(Sr.GetPixel(p,1).B -
Tr.GetPixel(p,Tr.Height -1).B);
                }
                valueR= valueR / 255; valueG= valueG / 255; valueB=
valueB / 255;
                puanlar[3]= Convert.ToInt32(1000 * (valueR +valueG
+valueB) / (3 * gen));
            }
```

```
            //final.Image= Tr;
            //final.Refresh();
           // this.Refresh();
            return puanlar;
        }

        private void buton26_Click(object sender,EventArgs e)
        {
            pictureSol1.Image= final.Image;
            ImageMatrix2= ImageMatrix1;
            int xx= -1,yy= -1;
            for (int y= 0; y<2 * mboyut -1; y++)
            {
                for (int x= 0; x<2 * mboyut -1; x++)
                {
                    if (matrix[x,y]== 1)
// 1: Doluların çevresini kontrol
                    {
                        if (xx<0) {xx= x;}
                        if (yy<0) {yy= y;}

                        Bitmap Sr= Kes((Bitmap)rekorPuzzle,x * gen,y
* yuk,gen,yuk);
                        Rectangle srcRegion= new
Rectangle(0,0,gen,yuk);
                        Rectangle hedefBolge= new Rectangle((x-
xx)*gen,(y-yy)*yuk,gen,yuk);

                        using (Graphics grafikrD=
Graphics.FromImage(pictureSol1.Image))
{grD.DrawImage(Sr,hedefBolge,srcRegion,GraphicsUnit.Pixel);}
//yeşile boya
                        ImageMatrix2[x -xx,y -yy]= Sr;
                    }
                }
            }

            pictureSol1.Refresh();
        }
```

```
        private void buton27_Click(object sender,EventArgs e)
        {
            final.Image=ParcalaCiz((Bitmap)final.Image,track.Value);
            final.Refresh();
        }
        public Bitmap ParcalaCiz(Bitmap input,int parca)
        {
            Bitmap result=new Bitmap(parca*gen,parca*yuk);
            result=(Bitmap)final.Image;

            for (int yx = 1; yx < (mboyut +1) -1; yx++)
            {
                for (int xx = 1; xx < (mboyut * gen) -1; xx++)
                {
                    int value= getValX(xx,y * yuk);
          result.SetPixel(xx,yx * yuk,Color.FromArgb(value,0,0));

                    //rekorPuzzle.GetPixel(x,y)
                }
            }


            for (int xx = 1; xx < (mboyut +1) -1; xx++)
            {
                for (int yy = 1; yy < (mboyut * yuk) -1; yy++)
                {
                    int value= getValY(xx * gen,yy);
                   result.SetPixel(xx *
gen,yy,Color.FromArgb(value,0,0));
                }
            }
            return result;
        }

        private void buton28_Click(object sender,EventArgs e)
        {
            pictureSol2.Image= final.Image;
            ImageMatrix3= ImageMatrix2;
```

```
            //Bitmap[,] ImageMatrix3= new
Bitmap[track.Value,track.Value];

            for (int y= 0; y< mboyut; y++)
            {
                for (int x= 0; x<mboyut; x++)
                {
                        Bitmap Sr= ImageMatrix2[x,y];
                        Rectangle srcRegion= new
Rectangle(0,0,gen,yuk);
                        Rectangle hedefBolge= new Rectangle(x* gen,y
* yuk,gen,yuk);
                        using (Graphics grafikrD=
Graphics.FromImage(pictureSol2.Image))
{grD.DrawImage(Sr,hedefBolge,srcRegion,GraphicsUnit.Pixel);}
//yeşile boya

                    int[] puanlar= Kontrol(Sr,x,y);
                    Console.WriteLine();
                    Console.Write(" Sol= " +puanlar[2]);
                    Console.Write(" Sağ= " +puanlar[0]);
                    Console.Write(" Alt= " +puanlar[1]);
                    Console.Write(" Ust= " +puanlar[3]);
                    int total= puanlar[0] +puanlar[1] +puanlar[2]
+puanlar[3];
                    Console.Write(" Total= " +total);

                    Font arialFont= new Font("Verdana",5);
                    PointF solKonum= new PointF(0,yuk/2);
                    PointF sagKonum= new PointF(gen-13,yuk / 2);
                    PointF altKonum= new PointF(gen / 2,yuk-8);
                    PointF ustKonum= new PointF(gen / 2,0);
                    using (Graphics grafikrD=
Graphics.FromImage(Sr))
{grD.DrawString(puanlar[2].ToString(),arialFont,Brushes.Red,solKonum
);} //Yazı Ekle
                    using (Graphics grafikrD=
Graphics.FromImage(Sr))
```

```
{grD.DrawString(puanlar[1].ToString(),arialFont,Brushes.Red,altKonum
);} //Yazı Ekle
                    //using (Graphics grafikrD=
Graphics.FromImage(Sr))
{grD.DrawString(puanlar[0].ToString(),arialFont,Brushes.Red,sagKonum
);} //Yazı Ekle
                   // using (Graphics grafikrD=
Graphics.FromImage(Sr))
{grD.DrawString(puanlar[3].ToString(),arialFont,Brushes.Red,ustKonum
);} //Yazı Ekle
                }
            }

            pictureSol2.Refresh();
        }
        public int[] Kontrol(Bitmap input,int xa,int ya)
        {
            double value= 0;
            Bitmap Sr= ImageMatrix2[xa,ya];
            Bitmap Tr= ImageMatrix2[xa,ya];
            int[] puanlar= {0,0,0,0 };// sağ-alt-sol-üst

            if (xa<mboyut -1) //sağ puan
            {
                value= 0;
                Tr= ImageMatrix2[xa +1,ya];
                for (int p= 0; p<= Tr.Height -1; p++)
                {
                    value += Math.Abs(Sr.GetPixel(Sr.Width -1,p).R-
Tr.GetPixel(0,p).R);
                    value += Math.Abs(Sr.GetPixel(Sr.Width -1,p).G-
Tr.GetPixel(0,p).G);
                    value += Math.Abs(Sr.GetPixel(Sr.Width -1,p).B-
Tr.GetPixel(0,p).B);
                }
                puanlar[0]= Convert.ToInt32(value/(3*Tr.Height));
            }

            if (xa>0)  //sol puan
```

```
            {
                value= 0;
                Tr= ImageMatrix2[xa -1,ya];
                for (int p= 0; p<= Tr.Height -1; p++)
                {
                    value += Math.Abs(Sr.GetPixel(0,p).R -
Tr.GetPixel(Tr.Width -1,p).R);
                    value += Math.Abs(Sr.GetPixel(0,p).G -
Tr.GetPixel(Tr.Width -1,p).G);
                    value += Math.Abs(Sr.GetPixel(0,p).B -
Tr.GetPixel(Tr.Width -1,p).B);
                }
                puanlar[2]= Convert.ToInt32(value / (3 *
Tr.Height));


                //for (int p= 0; p<= Tr.Height -1; p++)
                //{
                //    ImageMatrix3[xa,ya].SetPixel(Sr.Width -
1,p,Color.FromArgb(puanlar[2],0,0));
                //}
            }

            if (ya<mboyut -1) //alt
            {
                value= 0;
                Tr= ImageMatrix2[xa,ya +1];
                for (int p= 0; p<= Tr.Width -1; p++)
                {
                    value += Math.Abs(Sr.GetPixel(p,Sr.Height -1).R-
Tr.GetPixel(p,0).R);
                    value += Math.Abs(Sr.GetPixel(p,Sr.Height -1).G-
Tr.GetPixel(p,0).G);
                    value += Math.Abs(Sr.GetPixel(p,Sr.Height -1).B-
Tr.GetPixel(p,0).B);
                }
                puanlar[1]= Convert.ToInt32(value / (3 *
Tr.Height));
            }
```

```
            if (ya>0) //ust
            {
                value= 0;
                Tr= ImageMatrix2[xa,ya -1];
                for (int p= 0; p<= Tr.Width -1; p++)
                {
                    value += Math.Abs(Sr.GetPixel(p,0).R -
Tr.GetPixel(p,Tr.Height -1).R);
                    value += Math.Abs(Sr.GetPixel(p,0).G -
Tr.GetPixel(p,Tr.Height -1).G);
                    value += Math.Abs(Sr.GetPixel(p,0).B -
Tr.GetPixel(p,Tr.Height -1).B);
                }
                puanlar[3]= Convert.ToInt32(value / (3 *
Tr.Height));


                //for (int p= 0; p<= Tr.Height -1; p++)
                //{
                //    ImageMatrix3[xa,ya].SetPixel(p,Tr.Height-
1,Color.FromArgb(puanlar[3],0,0));
                //}
            }

            return puanlar;
        }

        public int[] Kontrol1(Bitmap input,int xa,int ya)
        {

            double value= 0;
            Bitmap Sr= input;
            Bitmap Tr= Kes((Bitmap)final.Image,0,0,gen,yuk);
            int[] puanlar= {0,0,0,0 };// sağ-alt-sol-üst


            if (matrix[xa+1,ya]== 1) //sağ puan
            {
```

```
                value= 0;
                Tr= Kes((Bitmap)final.Image,xa*gen +gen,ya *
yuk,gen,yuk);

                for (int p= 0; p<= Tr.Height-1; p++)
                {
                    value += Math.Abs(Sr.GetPixel(Sr.Width-1,p).R -
Tr.GetPixel(0,p).R);
                    value += Math.Abs(Sr.GetPixel(Sr.Width -1,p).G -
Tr.GetPixel(0,p).G);
                    value += Math.Abs(Sr.GetPixel(Sr.Width -1,p).B -
Tr.GetPixel(0,p).B);
                }
                puanlar[0]= Convert.ToInt32(value / (3 *
Tr.Height));

                //for (int p= 0; p<= Tr.Height -1; p++)
                //{
                //
ImageMatrix3[xa,ya].SetPixel(0,p,Color.FromArgb(puanlar[0] ,0,0));
                //}
            }

            if (matrix[xa -1,ya]== 1)  //sol puan
            {
                value= 0;
                Tr= Kes((Bitmap)final.Image,xa * gen -gen,ya *
yuk,gen,yuk);
                for (int p= 0; p<= Tr.Height -1; p++)
                {
                    value += Math.Abs(Sr.GetPixel(0,p).R -
Tr.GetPixel(Tr.Width -1,p).R);
                    value += Math.Abs(Sr.GetPixel(0,p).G -
Tr.GetPixel(Tr.Width -1,p).G);
                    value += Math.Abs(Sr.GetPixel(0,p).B -
Tr.GetPixel(Tr.Width -1,p).B);
                }
                puanlar[2]= Convert.ToInt32(value / (3 *
Tr.Height));
```

```
                //for (int p= 0; p<= Tr.Height -1; p++)
                //{
                //    ImageMatrix3[xa,ya].SetPixel(Sr.Width -
1,p,Color.FromArgb(puanlar[2],0,0));
                //}
            }


            if (matrix[xa,ya+1]== 1) //alt
            {
                value= 0;
                Tr= Kes((Bitmap)final.Image,xa * gen,ya * yuk
+yuk,gen,yuk);
                for (int p= 0; p<= Tr.Width -1; p++)
                {
                    value += Math.Abs(Sr.GetPixel(p,Sr.Height -1).R
-Tr.GetPixel(p,0).R);
                    value += Math.Abs(Sr.GetPixel(p,Sr.Height -1).G
-Tr.GetPixel(p,0).G);
                    value += Math.Abs(Sr.GetPixel(p,Sr.Height -1).B
-Tr.GetPixel(p,0).B);
                }
                puanlar[1]= Convert.ToInt32(value / (3 *
Tr.Height));


                //for (int p= 0; p<= Tr.Height -1; p++)
                //{
                //
ImageMatrix3[xa,ya].SetPixel(p,0,Color.FromArgb(puanlar[1],0,0));
                //}
            }

            if (matrix[xa,ya -1]== 1) //ust
            {
                value= 0;
```

```
                Tr= Kes((Bitmap)final.Image,xa * gen,ya * yuk -
yuk,gen,yuk);
                for (int p= 0; p<= Tr.Width -1; p++)
                {
                    value += Math.Abs(Sr.GetPixel(p,0).R -
Tr.GetPixel(p,Tr.Height -1).R);
                    value += Math.Abs(Sr.GetPixel(p,0).G -
Tr.GetPixel(p,Tr.Height -1).G);
                    value += Math.Abs(Sr.GetPixel(p,0).B -
Tr.GetPixel(p,Tr.Height -1).B);
                }
                puanlar[3]= Convert.ToInt32(value / (3 *
Tr.Height));


                //for (int p= 0; p<= Tr.Height -1; p++)
                //{
                //    ImageMatrix3[xa,ya].SetPixel(p,Tr.Height-
1,Color.FromArgb(puanlar[3],0,0));
                //}
            }

            return puanlar;
        }
        public int[] YerlesimPuani44(int xa,int ya)
        {
            // int[] array= new int[4];
//0:Sağında,1:Altında,2:Solunda,3:Üstünde
            double valueR= 0,valueG= 0,valueB= 0;
            //Bitmap Sr= new Bitmap(input); //Input Image Matrix[]
içinden parça x1,y1
            Bitmap Sr= Kes((Bitmap)final.Image,xa * gen,ya *
yuk,gen,yuk);
            Bitmap Tr= Kes((Bitmap)final.Image,0,0,gen,yuk);


            int[] puanlar= {0,0,0,0 };// sağ-alt-sol-üst

            if (xa<mboyut -1) //sağ puan
```

```
            {
                Tr= Kes((Bitmap)final.Image,xa * gen +gen,ya *
yuk,gen,yuk);
                for (int p= 1; p<= Tr.Height -1; p++)
                {
                    valueR += Math.Abs(Sr.GetPixel(Sr.Width -1,p).R
-Tr.GetPixel(1,p).R);
                    valueG += Math.Abs(Sr.GetPixel(Sr.Width -1,p).G
-Tr.GetPixel(1,p).G);
                    valueB += Math.Abs(Sr.GetPixel(Sr.Width -1,p).B
-Tr.GetPixel(1,p).B);
                }
                valueR= 1 +valueR / 255; valueG= valueG / 255;
valueB= valueB / 255;
                puanlar[0]= Convert.ToInt32(1000 * (valueR +valueG
+valueB) / (3 * gen));
            }
            if (xa>0)  //sol puan
            {
                Tr= Kes((Bitmap)final.Image,xa * gen -gen,ya *
yuk,gen,yuk);
                for (int p= 1; p<= Tr.Height -1; p++)
                {
                    valueR += Math.Abs(Sr.GetPixel(1,p).R -
Tr.GetPixel(Tr.Width -1,p).R);
                    valueG += Math.Abs(Sr.GetPixel(1,p).G -
Tr.GetPixel(Tr.Width -1,p).G);
                    valueB += Math.Abs(Sr.GetPixel(1,p).B -
Tr.GetPixel(Tr.Width -1,p).B);
                }
                valueR= valueR / 255; valueG= valueG / 255; valueB=
valueB / 255;
                puanlar[2]= Convert.ToInt32(1000 * (valueR +valueG
+valueB) / (3 * gen));
            }


            if (ya<mboyut -1) //alt
            {
```

```
                Tr= Kes((Bitmap)final.Image,xa * gen,ya * yuk
+yuk,gen,yuk);
                for (int p= 1; p<= Tr.Width -1; p++)
                {
                    valueR += Math.Abs(Sr.GetPixel(p,Sr.Height -1).R
-Tr.GetPixel(p,1).R);
                    valueG += Math.Abs(Sr.GetPixel(p,Sr.Height -1).G
-Tr.GetPixel(p,1).G);
                    valueB += Math.Abs(Sr.GetPixel(p,Sr.Height -1).B
-Tr.GetPixel(p,1).B);
                }
                valueR= valueR / 255; valueG= valueG / 255; valueB=
valueB / 255;
                puanlar[1]= Convert.ToInt32(1000 * (valueR +valueG
+valueB) / (3 * gen));
            }

            if (ya>0) //ust
            {
                Tr= Kes((Bitmap)final.Image,xa * gen,ya * yuk -
yuk,gen,yuk);
                for (int p= 1; p<= Tr.Width -1; p++)
                {
                    valueR += Math.Abs(Sr.GetPixel(p,1).R -
Tr.GetPixel(p,Tr.Height -1).R);
                    valueG += Math.Abs(Sr.GetPixel(p,1).G -
Tr.GetPixel(p,Tr.Height -1).G);
                    valueB += Math.Abs(Sr.GetPixel(p,1).B -
Tr.GetPixel(p,Tr.Height -1).B);
                }

                valueR= valueR / 255; valueG= valueG / 255; valueB=
valueB / 255;
                puanlar[3]= Convert.ToInt32(1000 * (valueR +valueG
+valueB) / (3 * gen));
            }
            //final.Image= Tr;
            //final.Refresh();
            // this.Refresh();
```

```
            return puanlar;
        }

        private void buton29_Click(object sender,EventArgs e)
        {
            pictureSol2.Image= final.Image;
            for (int yx = 0; yx < mboyut; yx++)
            {
                for (int xx = 0; xx < mboyut; xx++)
                {
                    Bitmap Sr= ImageMatrix3[xx, yx];
                    int[] puanlar= Kontrol(Sr, xx, yx);

                    Rectangle srcRegion= new Rectangle(0,0,gen,yuk);
                    Rectangle hedefBolge= new Rectangle(x * gen,y *
yuk,gen,yuk);

                    using (Graphics grafikrD=
Graphics.FromImage(pictureSol2.Image))
{grD.DrawImage(Sr,hedefBolge,srcRegion,GraphicsUnit.Pixel);}
//yeşile boya
                }
            }
            pictureSol2.Refresh();
        }

        private void buton24_Click(object sender,EventArgs e)
        {
            trans= new Bitmap(yedek.Width +1,yedek.Height +1);
            Graphics grafik= Graphics.FromImage(trans);
            grafik.Clear(Color.FromArgb(0,0,0,0));
            yedek.Image= trans;


            for (int jx= 0; jx<rekorPuzzle.Height; jx++)
            {
                for (int ix= 0; ix<rekorPuzzle.Width; ix++)
                {
```

```
                    if (rekorPuzzle.GetPixel(ix,jx)==
Color.FromArgb(255,255,0,0))
//Kırmızıyı bul
                    {

rekorPuzzle.SetPixel(i,j,Color.FromArgb(0,0,0,0));
//Trans yap
                    }
                }
            }

            //Bitmap tempRek;//= rekorPuzzle;
            // gg1.Clear(Color.FromArgb(0,0,200,0));

            Rectangle rect0= new Rectangle(0,0,gen * track.Value,yuk
* track.Value); //200x200
            Bitmap input= (Bitmap)rekorPuzzle;

            int xmin= 999,ymin= 999,x_maxx= 0,y_maxx= 0;
            for (int jx= 0; jx<rekorPuzzle.Height; jx++)
            {
                for (int ix= 0; ix<rekorPuzzle.Width; ix++)
                {
                  if (input.GetPixel(ix,jx)==
Color.FromArgb(255,255,0,0))
// ilk kırmızıyı bul
                    {
                        xmin= ix;
                        ymin= jx;
                    }
                }
            }
            //minX++;
            //maxX++;
            // minY++;
            // maxY++;

            xmin= minX * mX / track.Value;
            ymin= minY * mX / track.Value;
```

```
            x_maxx= maxX * mX / track.Value;
            y_maxx= maxY * mX / track.Value;

            width= x_maxx -xmin;
            height= y_maxx -ymin; // input size
            Rectangle srcRect= new
Rectangle(xmin,ymin,width,height);

            Rectangle hedefBolge= new Rectangle(0,0,width,height);
            using (Graphics grafikrD= Graphics.FromImage(trans))
{grD.DrawImage(input,hedefBolge,srcRect,GraphicsUnit.Pixel);}

            yedek.Image= trans;
            yedek.Refresh();
            this.Refresh();
        }

        private void buton16_Click_2(object sender,EventArgs e)
        {
            final.Image= rekorPuzzle;
            buton16.Text= TopRekor.ToString();
            final.Refresh();
            this.Refresh();
        }

        private void btnTemizle_Click(object sender,EventArgs e)
        {Temizle();
            mboyut= (int)Math.Round(Math.Sqrt(Lcand.Items.Count));
            btnBoyut.Text= mboyut.ToString() +"x"
+mboyut.ToString();

            trans= new Bitmap(yedek.Width,yedek.Height);
            Graphics grafik= Graphics.FromImage(trans);
            yedek.Image= trans;

            TopRekor= 999999;
            listBox1.Items.Clear();
            posX= 0; posY= 0;
        }
```

```
        private void btnAc_Click(object sender,EventArgs e)
        {

if(openFile.ShowDialog()==System.Windows.Forms.DialogResult.OK)
            {butun.Image=
ResizeImage((Bitmap)Bitmap.FromFile(openFile.FileName),mX,mY);}

            //parcali.Image=Parcala((Bitmap)butun.Image,4)[1,1];
            //Karistir(Parcala((Bitmap)butun.Image,4));
        }

        private void btnBoyut_Click(object sender,EventArgs e)
        {
            Sil(mX +" " +(mY -yuk) +" " +1);
        }

        private void btnPhoto_Click(object sender,EventArgs e)
        {
         //if (!CamCapture.Visible)
{CamCapture.Start(0);CamCapture.Visible=true;}
         //else {CamCapture.Stop(); CamCapture.Visible=false;}
        }

        //private void CamCapture_ImageCaptured(object
source,WebCam_Capture.WebcamEventArgs e)
        //{
        //    this.butun.Image= e.WebCamImage;
        //}

        #region
**************************************************************
Eskiler
**************************************************************
        public void Birlestir(int adet)
        {
            Bitmap[,] input= ImageMatrix1;
            Bitmap result= (Bitmap)final.Image;
```

```
            int xx0= 0,yy0= 0;
            Rectangle rect0= new Rectangle(0,0,gen,yuk);
            Rectangle hedefBolge= new Rectangle(mX,mY,gen,yuk);

            string[] xyy0;
            if (Lplaced.Items.Count>0) // Daha önce eklenmiş ise
ListFinal'in son elemanı=XYY0
            {
                xyy0= Lplaced.Items[Lplaced.Items.Count -
1].ToString().Split(null);
                xx0= int.Parse(xyy0[0]);
                yy0= int.Parse(xyy0[1]);

                for (int n= 2; n<adet; n++)
                {
                    int rekor= 0; int yn= 0; int rekX= 0; int rekY=
0;
                    for (int nn= 0; nn<Lcand.Items.Count; nn++)
                    {
                        string[] xy1=
Lcand.Items[nn].ToString().Split(null);
                        int x1= int.Parse(xy1[0]);
                        int y1= int.Parse(xy1[1]);
                        int[] s=
Benzerlik(input[xx0,yy0],input[x1,y1]);
                        if (s.Max()>rekor) {rekor= s.Max(); rekX=
x1; rekY= y1; yn= Array.IndexOf(s,s.Max());}
                    }

                    string yon= "";
                    switch (yn)
                    {
                        case 1:
                            yon= "Sağında: "; mX += gen;
                            break;

                        case 2:
                            yon= "Altında: "; mY += yuk;
                            break;
```

```
                        case 3:
                            yon= "Solunda: "; mX -= gen;
                            break;

                        case 4:
                            yon= "Üstünde: "; mY -= yuk;
                            break;

                    }

                    hedefBolge= new Rectangle(mX,mY,gen,yuk);
                    using (Graphics grafikrD=
Graphics.FromImage(result))
{grD.DrawImage(input[rekX,rekY],hedefBolge,rect0,GraphicsUnit.Pixel)
;}
                   // matrix[mX / gen ,mY / yuk ]= 1;
                    if (Lcand.Items.Contains(rekX +" " +rekY))
{Lcand.Items.Remove(rekX +" " +rekY); Lplaced.Items.Add(rekX +" "
+rekY);}
                    final.Image= result;
                    final.Refresh();
                }
            }
            else
            {
                xyy0= Lcand.Items[0].ToString().Split(null);
                xx0= int.Parse(xyy0[0]);
                yy0= int.Parse(xyy0[1]);
                Lplaced.Items.Add(Lcand.Items[0]);
                Lcand.Items.RemoveAt(0);
                using (Graphics grafikrD=
Graphics.FromImage(result))
{grD.DrawImage(input[xx0,yy0],hedefBolge,rect0,GraphicsUnit.Pixel);}
                //matrix[mX / gen,mY / yuk]= 1;
            }
        }
        #endregion
    }
}
```