

**“LUCIAN BLAGA” UNIVERSITY OF SIBIU
ENGINEERING FACULTY
DEPARTMENT OF COMPUTER SCIENCE, ELECTRICAL
AND ELECTRONICS ENGINEERING**

Dissertation

**SCIENTIFIC ADVISOR:
Prof.univ.dr.eng. Remus BRAD**

**GRADUATE:
Bogdan-George Herțoiu
Advanced Computing Systems**

**“LUCIAN BLAGA” UNIVERSITY OF SIBIU
ENGINEERING FACULTY
DEPARTMENT OF COMPUTER SCIENCE, ELECTRICAL
AND ELECTRONICS ENGINEERING**

Automatic detection of new classes of objects in images

SCIENTIFIC ADVISOR:
Prof. univ. dr. eng. Remus BRAD

GRADUATE:
Bogdan-George Herțoiu
Advanced Computing Systems

Table of Contents

1. Introduction	1
2. Abstract.....	4
3. Theoretical substantiation.....	5
3.1. Image processing	5
3.2. Object detection	7
3.3 Distance distribution.....	9
3.4 Hierarchical clustering.....	9
4. Development of the subject	12
4.1. Methodology.....	12
5. Results	31
6. Conclusions	34
7. References	35

1. Introduction

According to IBM [14], computer vision represents a domain of the artificial intelligence that enables the computers to gather information from the images and process them. This process is similar to the human eye vision. The strength of the human vision is the ability to learn from new experiences and classify objects in a short period of time, such as what kind of object it is, how far it is and much more.

Computer vision is used in many domains of activity such as industrial automation, robotics, text analytics, motion detection etc. One of the domains focused on the research is the identification of objects in an image, using a camera and classifying the information. Different chess pieces will be used in a black background.

The basic functionality of computer vision is that it requires a good amount of data to learn, in order to be able to make distinctions and recognize objects. In this paper, images with chess pieces will be used and the program will identify the chess pieces, crop the sections individually and save them locally. Afterwards, the saved images will have normalized pixel values as well as their angle orientation.

Machine learning is a field of study in artificial intelligence that uses algorithms for learning the statistics of the data and performing tasks without explicit instructions [54].

This research holds an importance in the computer vision field. Creating an autonomous process to recognise different objects and creating labels for easier identification can help with the understanding of the objects present in the images and potentially be better than the human's abilities.

In this paper, there will be discussions about the fields of image processing, object detection, distance distribution and hierarchical clustering, since there are important subjects in the development of the research. Afterwards, the methodology of the research is presented, the tools used and presenting its conclusions, results and final thoughts.

Technologies that will be used are Python and clustering with the following libraries: OpenCV, Numpy arrays, Pandas dataframes, Matplots and Scipy.

The research was done using the language Python. According to Simon Yuill and Harry Halpin [42], Python is a powerful, elegant programming language that is easy to read and to understand. Unlike natural languages such as English, formal languages like Python strive to eliminate ambiguity. They have to eliminate ambiguity, because the code is supposed to tell the computer exactly what to do. The variables in Python have values that vary over the time of the running of a program. What is advantageous about variables is that they keep their value until

changed. What most people think of as “data” is generally stored in variables or collections of variables. Majority of programming is moving values around variables and then processing them according to the authors.

Some of the variables used in this research are arrays, dataframes, strings and floating/integer values. File manipulation using instructions designed specifically for image processing are used in this research, as well as for processing them. Another functionality of Python used that is mentioned by Simon Yuill and Harry Halpin [42], are modules which are Python scripts containing a set of classes and functions. These are useful to pack them up, and these allow to extend the language by importing new modules. The anatomy of a module is made of the import statements, to include the module in the environment, the module variables, the methods that can be called from module and objects. The modules in the research will be explained in the methodology section.

Guido van Rossum [48] highlights some of the advantages that Python offers. These are the following:

- the high-level data types allow expression of complex operations in a single statement
- statement grouping is done by indentation instead of beginning and ending brackets;
- no variable or argument declarations are necessary

Besides the programming language used, the important domains taken into consideration in this research are image processing, object detection, distance distribution and hierarchical clustering.

Acharya T, Ray AK. [1] talk about image processing growing in demand for many real life application areas, like pattern recognition, multimedia computing etc. It is an important step for processing images because it enables the computer to be able to read the images properly and process them, due to the complexity of shapes, colors and textures of the images. This process is a simplified representation for the computer to be able to read the images easier.

Once the images are better understood by the computer, having more precise and detailed object recognition becomes more important. Christian Szegedy, Alexander Toshev and Dumitru Erhan [47] highlight that not only is it important to classify images, but also about precisely estimating the class and location of objects contained within the images, also known as object detection.

In order to find correlation between multiple images, distance distribution is able to help to provide insights by comparing the distances between multiple images. The closer they are, the

more likely they are related which helps to identify similar images and group them into clusters and in this research, hierarchical clustering is used.

For the purpose of this research, the complexity of object detection has been reduced: a simple black background and recognizable objects have been used for easier processing and detection of the objects in order to be able to create the classifications, because the process of detection is not the main objective for this research.

2. Abstract

The human eye and the computer vision applications have in common the ability to observe the environment and learn from it. They take note of the particularities in the environment and make observations about the world. Humans can easily learn the traits of the objects from the background and find relation with the environment. With this innate ability, humans can gain experience and knowledge by observing and learning and use the knowledge to recognise future interactions. In other words, the process is to recognize the new objects first, recall the knowledge learnt and deduce what the object is in the end.

In this paper, different chess pieces will be used in a black background. Chess pieces that are used are pawns, bishops, rooks, knights, kings, queens. Each image with the black background can have a different amount of chess pieces. These images were captured via phone camera. The program reads each image and tries to identify the pieces present in the image and creates sections that can be cropped. The sections of the chess pieces are identified and cropped. The cropped images are saved locally in order to be processed. The next step is to normalize the pixel and angle values of the cropped pictures. The normalized images are saved locally as well. Afterwards, the program calculates the black pixel distribution of the normalized images in order to create matrices of similarities for each image and their values are saved in CSV files. Each image will be tested to see if they are similar and create a label for the program to identify future images, by using clustering methods, more specifically, hierarchical clustering. Additional parameters are used to model the dimensions of clusters in hierarchical clustering.

After the algorithm is finished, results will be shown regarding the number of clusters generated and how many images are in the clusters and how accurate the recognition was.

With these methods, the program will be able to recognise and create name labels for each chess piece that identifies in the input images.

3. Theoretical substantiation

In this chapter, it will describe the theoretical bases of the methods and procedures applied in the development of the dissertation, as well as the presentation of the technologies used, specifically, image processing, object detection, distance distribution and hierarchical clustering.

3.1. Image processing

According to the book Image Processing in C, by Dwayne Phillips [40], Image processing is responsible for manipulating I/O images, according to the author's desired manner and outputting the results. The first step is obtaining an image in a readable format. Its file format specifies how the information is stored. Some examples of file formats are: JPG, PNG. Once the image is obtained, the program needs to read it so it can be processed and written back to a file. Further on, Images are composed of pixels and can be seen as an array with rows and columns. The array values are RGB values, which stands for Red Green and Blue, ranging between 0 and 255. The simplest image representation of an image is grayscale images, because the RGB values are the same according to Dwayne Phillips [40].

One of the first mentions in image processing is the first electronic computer that could be programmable, ENIAC, built by Eckert and Mauchy in 1946, mentioned by Ernest L. Hall [5]. The company HP tech [13] specifies that ENIAC stands for Electronic Numerical Integrator and Computer and its first usage was to compute the values of artillery range. Many computer picture processing techniques have been developed. Ernest L. Hall [5] emphasizes that the domain of image processing was able to advance thanks to availability of image scanning and display hardware at a reasonable price, and the relatively free use of computers.

Another achievement of image processing was the race to the moon thanks to the American Jet Propulsion Laboratory (JPL). Chintan P. Dave, Rahul Joshi, S. S. Srivastava [6] found out that image processing techniques were used such as:

- geometric correction, responsible for removing geometrical distortions in an image
- gradation transformation, for enhancing the contrast values, due to nonuniform pixel intensity distribution, according to Jian Lu and Dennis M. Healy, Jr [22].

The noise removal on moon photos sent back by the Space Detector Ranger 7 in 1964, taking into account the position of the Sun and the environment of the Moon, according to Ernest L. Hall [5] [50].

Ernest L. Hall [5] mentions that computer processing techniques have also contributed to this development. Without the popularization of the fast Fourier transform algorithm by Cooley

and Tukey and others in 1965, transform processing of images would probably have remained in the domain of optics.

With the rise of image processing, problems can arise in creating classifications. The authors consider five of the major problems that have evolved in computer picture processing according to Ernest L. Hall [5]: enhancement, communications, reconstruction, segmentation, and recognition. These problems will remain important in most image processing applications.

Some of the work done in the image processing domain is done by Ozcanli, Ozge C., et al. [36] in the Vehicle model recognition using geometry and appearance of car emblems from rear view images. The authors augment shape with appearance in cases where segmentation is difficult and not sufficiently detailed or reliable to be categorized. It recognises vehicle categories from aerial videos, where it is necessary to discriminate intra category variations from inter category variations. The visual differences between cars of different and same types are subtle when the image capture is done from a high magnitude, thus, it is difficult. For example, qualitatively, the difference between a pickup truck and an SUV is not significantly more than the difference between two pickup trucks. Augmenting shape with appearance can improve recognition rate from poor contours or good contours that are not diagnostic. They propose to find correspondences between the two images based on shape only, then find similarities between the two images based on appearance at corresponding points. In this way, shape at first is used and appearance afterwards. Shape and appearance are used together to make the matching process complete.

Goswami D, Gaur R. [11] focuses on making an automatic license plate recognition system with Histogram Graph Algorithm, that uses character recognition on images to read vehicle plates in the form of segmented characters with Matlab. This is useful for car identification, such as automatic toll collection, traffic law enforcement, parking lot access control, and road traffic monitoring. The system recognizes a vehicle's license plate number from an image taken by a digital camera. Algorithm of the automatic license plate recognition system is fulfilled by techniques of image processing such as preprocessing, license plate detection, and character recognition. It is assumed that the Histogram algorithm works on images that have been captured from a fixed angle parallel to the horizon in different luminance conditions. It is also assumed the vehicle is stationary and images are captured at fixed distance.

3.2. Object detection

Image processing is one of the first steps to ensure that the data has the optimal parameters. By applying techniques to improve the quality of image and remove distortions. The next process is the Object detection, responsible for identifying the object in an image and classifying them.

Kenneth Marino, Ruslan Salakhutdinov, Abhinav Gupta [24] noticed an important common characteristic between computer vision applications and human eyes, which is the ability to acquire knowledge of the surroundings and use the knowledge to make observations about the world. Another thing that humans can do is to learn characteristics of the object and its relation to the environment. Humans gain experience and knowledge from learning and use it to recognise and interact with the environment. In other words, the process is to recognize the new object first, recall the knowledge learnt and finally, deduce what the object is.

Lubna Azis et. al. [2] talk about object detection being a combination of image classification with precise object localization that provides a complete and proper understanding of the image. In the past, manual feature extraction followed by shallow trainable architectures was used for object detection. With the introduction of deep learning tools, limitations of traditional object detection techniques were overcome that can learn semantic and deep level features. With these advancements, object detection has been divided in subcategories such as face detection, pedestrian detection and skeleton detection, etc. It is a fundamental computer vision process that provides detailed semantic information of image and video.

Object detection has two operations that are mentioned by Lubna Azis et. al. [2]: object localization and classification. The localization determines the location of an object in the image and the classification determines the category of the object. A problem with localization in object detection is due to occlusions, significant variations in viewpoints, scales and lighting.

There have been many methods developed for increasing the accuracy of object detection models. An example is The Viola-Jones framework, which became popular due to its successful application in the face-detection problem, and was later applied to different models such as pedestrian and car detections, according to Modesto Castrillón et. al. [3]. Aside from Viola-Jones framework, different approaches have focused on this ability but only open source implementations have been used by researchers. The OpenCV community has public domain classifiers for the face detection scenario and this framework is one of them [37].

The era of computer vision inventions begins with the development of the Deep Neural Network (DNN), according to Lubna Azis et. al. [2]. DNN works differently from the traditional approach due to more in-depth architecture, it can learn sophisticated features, and has a robust

training algorithm that allows features to learn the informational representation of objects without manually designing them.

The era of the neural network began in 1940; the basic idea behind it was to solve the common problem of learning by mimicking the human brain. The popularity of deep-learning increased in the late 1980s and 1990s with the development of a backpropagation algorithm proposed by Hinton [2].

In early 2000, Lubna Azis et. al. [2] observe that the popularity of deep learning began to decline due to a lack of big data, high computational power requirements, and performance insignificance as compared to other machine learning tools. The rise in popularity of Deep learning began in the year of 2006 with speech recognition.

The development of Convolutional neural networks (CNN) and GPU-accelerated deep-learning frameworks enabled the object detection algorithms to be created. Convolutional Neural Network (CNN) is one of the most important networks in the field of deep learning. It was invented by Lann LeCun in the 1990s [46]. CNN made significant achievements in areas like computer vision and natural language processing. For example, in Computer Vision, it can accomplish face recognition, autonomous vehicles.

Since the invention of R-CNN (region CNN), created by Ross Girshick a significant number of different and improved models have been proposed in the field of object detection such as Fast R-CNN, which improves the object detection task by combining Bounding Box regression and classification task, according to Dhruv Parthasarathy [7]. In contrast, Faster R-CNN generates a region proposal using the additional sub-network, while using fixed grid regression in YOLO (You Only Look Once) for object detection tasks. All of these object detection algorithms make real-time object detection more achievable by providing a better and more accurate way to identify objects on a basic R-CNN. Object detection models are very effective across different application domains such as face detection, generic object detection and pedestrian detection [2].

Zewen Li, Wenjie Yang, Shouheng Peng, Fan Liu [21] talk about world wide competitions, such as Voc Pascal Challenge, COCO, ImageNet Object Detection Challenge and Google Open Images Challenge. They have as their best object-detection algorithms methods inspired by CNN [37].

3.3 Distance distribution

In image processing and computer vision, the comparison of distributions is a frequently used technique. Some applications for these measures are image retrieval, classification, and matching systems. For these, the distributions could represent low-level features like pixel's intensity level, color, texture or higher-level features like objects. The comparison could be done using a unique feature, for example, the texture, or combining features in a multi-dimensional distribution as the fusion of color and texture features. In the field of medical imaging, comparing distributions are useful to achieve image registration, which is used for transforming different sets of data into one coordinate system [53]. More general applications such as object tracking, also use the comparison of distributions.

According to Jing Li, Bao-Liang Lu [20], measuring the distance or similarity between images is an important problem in computer vision. It is suggested that humans judge image similarity in a nonmetric way. As in the field of computer vision, the most commonly used distance is Euclidean distance, which converts images into vectors according to gray levels of each pixel, and then compares intensity differences pixel by pixel.

Eric Nowak, Frédéric Jurie [31] found out that most of the contributions consist in finding a function mapping the feature space into a target space such that a simple distance can eventually be used in the target space.

With similarity measures, data samples are grouped into different classifications. For instance, Mahmoud Harmouch [23] talks about KNN that can be used for classifications, where the data objects are labeled based on the features' similarity. The similarity measure is usually expressed as a numerical value: in this paper, the lower value data samples have, the more are alike.

3.4 Hierarchical clustering

To group data, Murtagh F and Contreras P. [29] suggest to measure the elements and calculate their distances relative to each other in order to observe which elements belong to a group.

Hierarchical clustering is a technique for performing data exploratory analysis. It is an unsupervised technique. Hierarchical clustering consists in building a binary merge tree, starting from the data elements stored at the leaves (interpreted as singleton sets) and proceed by merging two by two the "closest" sub-sets (stored at nodes) until the root of the tree is reached that contains

all the elements of data set X . It is denoted by $\Delta(X_i, X_j)$ and it is the distance between any two subsets of X , called the linkage distance [30]. The graphical representation of hierarchical clustering is called a dendrogram. They can be represented by Venn diagrams or binary merge trees.

To further understand how hierarchical clustering works, the following figure 3.1 shows how a dendrogram looks like and the process behind clustering:

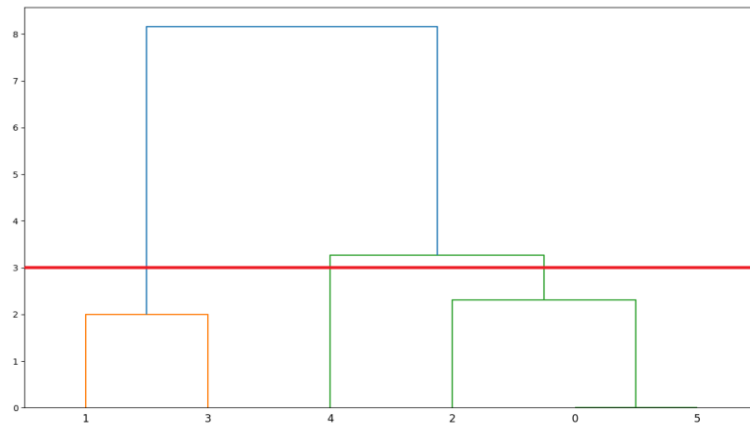


Figure 3.1 - Dendrogram example for hierarchical clustering

In this figure, we have several objects from an array. The array values are [3,7,1,9,5,3]. The dendrogram is formed by connecting the closest objects, all the way until no more connections can be done. A distance threshold is used to determine how many clusters are formed, marked as a red line.

There are several mentions according to the authors [55], regarding hierarchical clustering: Gordon (1981) in the Classification book [9], March (1983) related to Rearrangement Clustering [4], Jain and Dubes (1988) in Algorithms for Clustering Data [17], Gordon (1987) [gordon last page], Mirkin (1996) in the book Mathematical Classification and Clustering [27], Jain, Murty and Flynn (1999) in Data Clustering: A Review [18], Xu and Wunsch (2005) in Survey of clustering algorithms [28].

Fionn Murtagh and Pedro Contreras [28] made a survey about the hierarchical clustering: Lerman and Janowitz in 1981 and 2010 respectively, present reviews of clustering, as well as lattices that generalize trees. Hierarchical clustering in information retrieval was made by van Rijsbergen in 1979 which became an important contributor and was continued in the work of

Willett and coworkers in 1984. Many other mathematical views of hierarchy, are included in Murtagh from 2017.

Before making use of clustering, data needs to be measured. To group data, it is required to find a way to measure the distance of elements to be able to decide which elements belong to a group according to Fionn Murtagh and Pedro Contreras [28]. In other words by comparing the similarity, it is possible to observe which data are close to each other.

Fionn Murtagh and Pedro Contreras [28] highlight the importance of hierarchical clustering and the applications are the following: data analysis, interactive user interfaces, and pattern recognition. One basic motivation for using hierarchical clustering is to have a large number of partitions. Each partition is associated with a level of the hierarchy, its dendrogram representation, or, as a mathematical graph theory term, a binary, rooted tree.

There are several advantages using hierarchical clustering, compared to other clustering methods [8]:

- It handles non-convex clusters and clusters of different sizes and densities.
- It handles missing data and noisy data.
- It can reveal the hierarchical structure of the data, which can be useful for understanding the relationships among the clusters.

The disadvantages of hierarchical clustering are the following [8]:

- A criterion to stop the clustering process and determine the final number of clusters.
- The computational cost and memory requirements of the method can be high, if datasets with a large amount of data are used.
- Linkage criterion's initial conditions can affect the results, which determines the distance to use between sets of observation. The algorithm will merge the pairs of clusters that minimize this criterion and distance metric used according to scikit-learn [43].
- This method can handle different types of data and reveal the relationships among the clusters. However, it can have high computational cost and results can be sensitive to some conditions.

4. Development of the subject

In this chapter, it will be discussed the tools used for developing the research, offering detailed description on how it was achieved, as well as the flow of the program.

4.1. Methodology

In this paper, I am using a dataset of images that contain chess pieces in a black background. Each image can have different amounts of chess pieces, which have been captured by using a smartphone. The dataset contains over 40 pieces identified by the method that will be described further in detail.

The language used to create the program is called Python. Python is an interpreted, general-level programming language, using indentation to create code readability. Python uses whitespace indentation, rather than parentheses or keywords, to delimit blocks. Thus, the visual structure of the program is accurately represented and the semantic structure of the program is clear. The language structure, as well as its object-oriented approach, are intended to help programmers write clear and logical code for projects, no matter how big they are. Python aims for a simpler, less cluttered syntax and grammar, while giving developers a choice in their coding methodology.

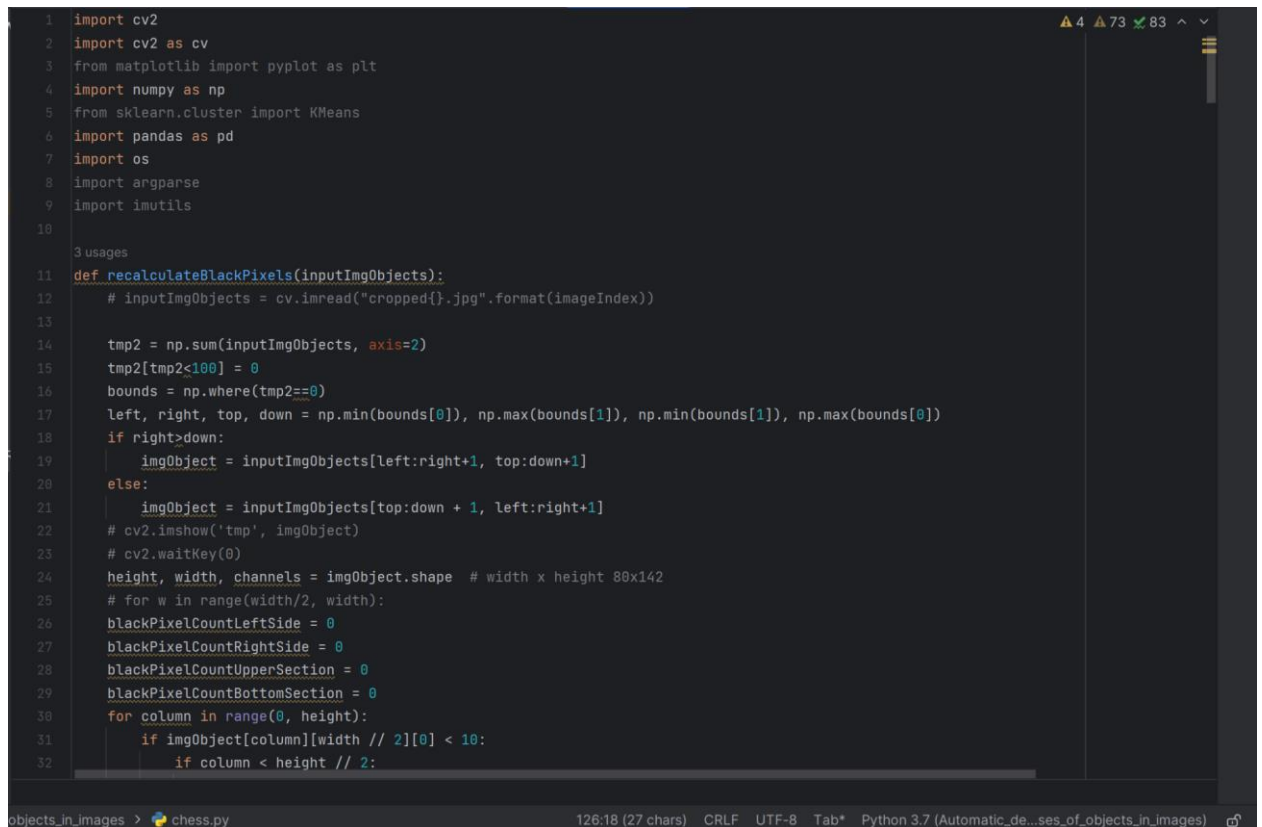
Python is a dynamic and garbage-collector language and it supports multiple programming paradigms, including structured, object-oriented, and functional programming. It contains a large number of libraries. Python's standard library, one of the language's most advantageous, provides tools suitable for many tasks. It includes modules for creating graphical interfaces, image processing and detection and much more.

Libraries such as NumPy, SciPy, and Matplotlib enable efficient use of Python in scientific computing, with specialized libraries such as Biopython and Astropy providing domain-specific functionality. Python is commonly used in artificial intelligence applications and machine learning projects with the help of libraries such as TensorFlow, Keras, Pytorch and Scikit-learn.

Python's functionalities, statements, data types, and syntax provide many benefits and simplicity for the programmer. There are various basic data types, such as integers, floats, booleans, strings, etc., which can be dynamically assigned to a variable when declaring it without implicitly writing the data type used. Once a variable is assigned a value such as of type int, that variable will be of type int.

Python allows for multiple value assignments, incrementing and decrementing values, conversion operations between various data types, indexing for subsequences of lists, and more. The arithmetic operators used are arithmetic, logical, and the evaluation of the expressions is done on the right side of the assignment, and the assignment of values is done in order, from left to right.

The development of the program was also done in JetBrains Pycharm, shown in Figure 4.1 which represents an IDE (Integrated Development Environment), used in computer programming, for the Python language, developed by the Czech company JetBrains. It offers code analysis, a graphical debugger, an integrated tester, integration with version control systems (VCS). PyCharm can be used on multiple platforms, with Windows, macOS, and Linux versions. The reason for choosing this development environment is the convenience it offers for the programming environment and the multitude of functionalities available for Python [19].

The image shows a screenshot of the JetBrains PyCharm IDE. The main editor window displays a Python script with the following code:

```
1 import cv2
2 import cv2 as cv
3 from matplotlib import pyplot as plt
4 import numpy as np
5 from sklearn.cluster import KMeans
6 import pandas as pd
7 import os
8 import argparse
9 import imutils
10
11 3 usages
12 def recalculateBlackPixels(inputImgObjects):
13     # inputImgObjects = cv.imread("cropped{}.jpg".format(imageIndex))
14
15     tmp2 = np.sum(inputImgObjects, axis=2)
16     tmp2[tmp2<100] = 0
17     bounds = np.where(tmp2==0)
18     left, right, top, down = np.min(bounds[0]), np.max(bounds[1]), np.min(bounds[1]), np.max(bounds[0])
19     if right>down:
20         imgObject = inputImgObjects[left:right+1, top:down+1]
21     else:
22         imgObject = inputImgObjects[top:down + 1, left:right+1]
23     # cv2.imshow('tmp', imgObject)
24     # cv2.waitKey(0)
25     height, width, channels = imgObject.shape # width x height 80x142
26     # for w in range(width/2, width):
27     blackPixelCountLeftSide = 0
28     blackPixelCountRightSide = 0
29     blackPixelCountUpperSection = 0
30     blackPixelCountBottomSection = 0
31     for column in range(0, height):
32         if imgObject[column][width // 2][0] < 10:
33             if column < height // 2:
```

The interface includes a top toolbar with icons for file operations, a right sidebar for project structure, and a bottom status bar showing the current file path, character count, and encoding.

Figure 4.1 - JetBrains Pycharm interface

The useful features of this IDE are the following [9]:

- Coding support and analysis, with code completion, syntax and error highlighting, and quick fixes;

- Project and code navigation: specialized project views, file structure views and quick jump between files, classes, methods and usages;
- Python refactoring: includes renaming, method extraction, variable casting, constant casting, and more;
- Integrated Python debugger;
- Integrated unit testing with line-by-line code coverage;
- Version control integration: unified user interface for Mercurial, Git, Subversion, Perforce and CVS with changelists and merge;
- Support for scientific tools such as matplotlib, numpy and scipy;

The programming language used is Python alongside the following libraries: CV2, Matplotlib, Numpy, Os, Pandas and Scikit-learn and Scipy.

CV2, also known as OpenCV, which stands for Open Source Computer Vision Library, is a module, popular within the community, due to being open source and it suitable for computer vision and machine learning applications and it is a starting point for creating a framework for the computer vision domain. The module has many optimized algorithms for computer vision and machine learning, no matter if they are classic algorithms or up to date. These have many uses for these domains such as detection and face recognition, identifying objects and extracting their model, finding similar images from a database and many more. The library is used widely in companies and research groups. Popular companies like Google, Yahoo, Microsoft, Intel, IBM make use of this library. Practical usages for include stitching street viewing images together, detecting intrusions in surveillance video, monitoring mine equipment, helping robots navigate and pick up objects in a garage, detection of swimming pool drowning accidents, checking runways for debris, inspecting labels on products in factories around the world and face detection. It has C++, Python, Java and MATLAB interfaces and supports operating systems like Windows, Linux, Android and Mac OS [33].

Matplotlib is a comprehensive library for creating visualizations in different manners, in Python, thanks to J. D. Hunter [16]. The library can create plots, make interactive figures that can zoom, pan, update, customize visual style and layout, export to many file formats and use a rich array of third-party packages built on Matplotlib. It is independent of Matlab, and can be used in a Pythonic, object-oriented way. The Matplotlib developer community develops, maintains, and supports Matplotlib and its extensions to provide data visualization tools for the Scientific Python Ecosystem. Matplot is able to support the community with interactive data exploration, high-

quality raster and vector format outputs suitable for publication, graphical user interface and support embedding in applications and easy common plots as well as complex visualizations [16].

NumPy represents a module, useful for scientific computing in Python, that provides a multidimensional array object, and many methods for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, and much more. The Numpy's object that has the biggest influence, is the ndarray object. The ndarray contains n-dimensional arrays of homogeneous data types, with performance taken into account by using operations for performance and compiling them with efficiency. By using ndarray, NumPy can support an object-oriented approach. For example, ndarray is a class and it consists of methods and attributes. NumPy arrays enables the usage of advanced mathematical on large amounts of data. Numpy arrays have shown to be more efficient at handling data than the traditional Python arrays [32].

Os is a module that provides a portable way of using operating system dependent functionality [41]. It contains methods for interacting with the operating system, like creating files and directories, management of files and directories, input, output, etc [49]. This is useful for gathering the input files from folders in order to read them.

Pandas is an open source module, suitable for practical, real world data analysis. It is a fast and efficient DataFrame object for data manipulation with integrated indexing and is useful for working with tabular data, such as data stored in spreadsheets or databases. It supports the integration with many file formats or data sources out of the box (csv, excel, sql, json, etc.) [38].

According to Gavin Hackeling [12], machine learning is the design and study of software applications that use experiences from the past to make a decision in the future and learn from data acquired. The goal of machine learning is to generalize, or to make use of an unknown rule from examples of the rule's application. In other words, the application learns without being told. Scikit-learn was introduced in 2007 and thanks to its contributions, it has managed to be one of the most well made libraries, dedicated for machine learning. Scikit-learn provides algorithms for machine learning tasks such as classification, regression, dimensionality reduction, and clustering. It also provides modules for extracting features, optimizing, and evaluating models. Scikit-learn is built on the popular Python libraries NumPy and SciPy. NumPy offers extended support for efficient operations on large arrays and this module is also used in this paper. Gavin Hackeling [12] mentions that scikit-learn is suitable for academic research thanks to the API documentations as

well as being easy to model and experiment with different algorithms and parameters. In this paper, SciPy is mainly used for the process of hierarchical clustering.

In the following figure 4.2, it is presented a flowchart of the functionality of the program. The program starts reading the image files from the folder and applies image normalization for each file. Afterwards, the normalized image is processed and objects are detected and saved. The program counts how many black pixels are on each side of the image, in order to see how the pixels are distributed and find the right angle to rotate the image. This ensures that the objects have rotation invariance.

With the normalized and rotated images, they are all put together in a dataframe for easier organization and finally, the hierarchical clustering can be used for classifying the images obtained so far.

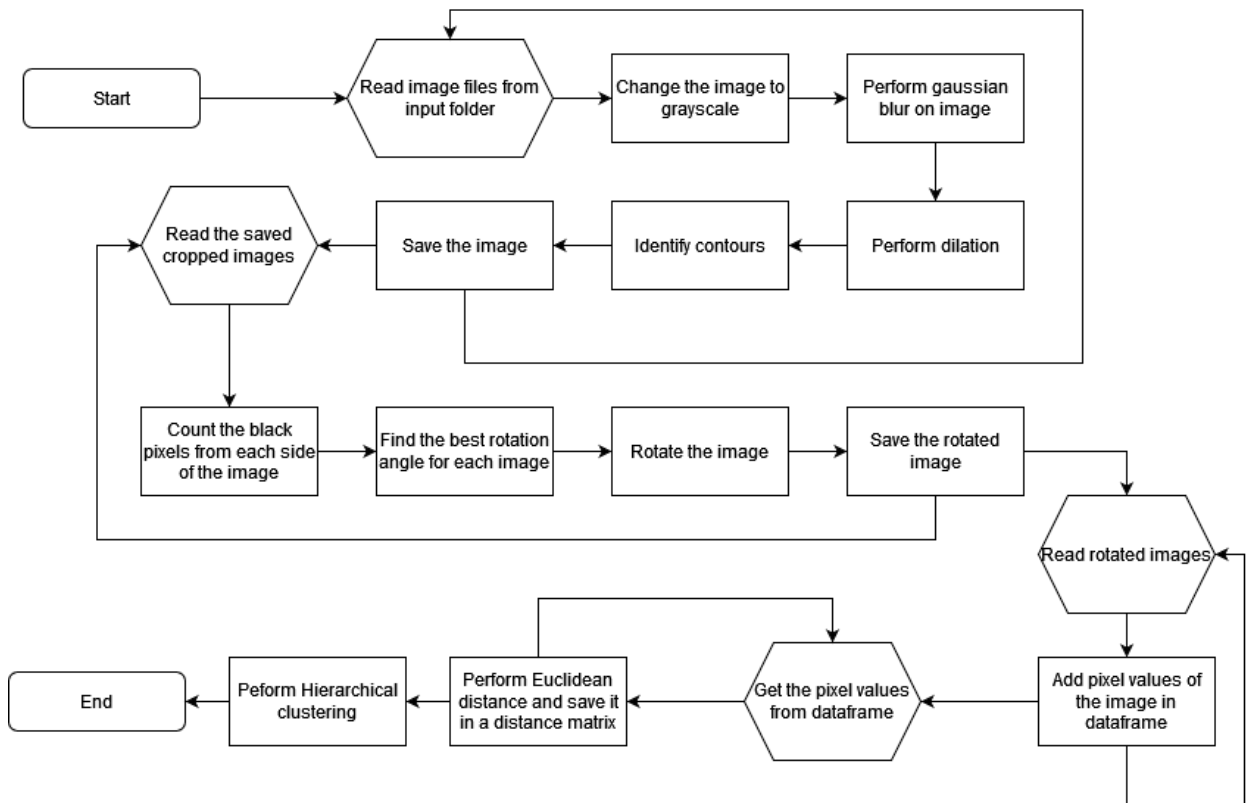


Figure 4.2 - Flowchart of the program

The pictures are located in a folder called “InputPictures”. An example of image is shown in Figure 4.3. The list of their names are taken by using listdir method from os module. The listdir takes every name of the folder as strings and puts them in an array for easier access of each file.

Every file from the array is read by using the cv method imread in a loop and every image goes through a process called normalization, which is necessary to remove any distortions present in the image so that the program can work with it. [39]. Normalization process of the pixel values is composed of these steps:

- using grayscale on image
- blurring with Gaussian blur
- dilation



Figure 4.3 - Example of input image

The purpose of grayscale is to convert the RGB pixel values in black and white. This is useful for removing color information and has different shades of gray, the brightest being white and darkest being black, making the computational requirements easier according to Isahit [15]. This is achieved by using the CV method cvtColor, which converts the current image to another color scheme, depending on the argument and in this case, it is COLOR_BGR2GRAY, that states the image to be in grayscale.

The next step of normalization is to use a blur method, called Gaussian blur. It is used for reducing image noise and detail [52]. This is achieved by using the cv method GaussianBlur. It uses a Gaussian function and it needs the width and height of the function specified which should

be positive and odd. According to OpenCV documentation, it should also be specified the standard deviation in the X and Y directions, sigmaX and sigmaY respectively. If only sigmaX is specified, sigmaY is taken as the same as sigmaX. If both are given as zeros, they are calculated from the function size [35].

Finally, through dilation, it adds pixels to the boundaries of objects in an image and the number of pixels added from the objects in an image depends on the size and shape of the structuring element used to process the image [25]. Dilation is considered a morphological operation and it is a set of operations that process images based on shapes according to Mathworks. Morphological operations apply a structuring element to an input image and generate an output image. This operation consists of convolving an image with kernel, which can have any shape or size. The kernel is normally centered and it can be identified by the anchor point. The anchor point is used for computation, while the the kernel is scanned over the image. By scanning, the maximal pixel value overlaps by the kernel and replaces the image pixel in the anchor point position with that maximal value. With this maximizing operation, it can be observed why it is called dilation. This is because the bright regions of the images are growing, thanks to dilation [34].

With grayscale, Gaussian blurring and dilation, the image is normalized. The next step is to find the actual chess pieces in the input image. This is achieved by using the cv method of finding contours. Contours are a curve joining all the continuous points (along the boundary), having the same color or intensity. The contours are useful for analysing the shape of the object and using it for detection and recognition. This method has better accuracy of detecting objects if the images are binary, hence why the input images have gone through the normalization process. A condition before finding the contours is to ensure that the background is black and the objects are white. Once the contours are found, they are drawn and its shape coordinates and sizes are taken into account in order to save the section of the image, the chess piece itself for further processing, as shown in Figure 4.4 for a rook piece.



Figure 4.4 - Rook image after the normalization process

The functionality of the code can be observed in the algorithm 4.1:

```
image_number = 0

inputDirectory = os.listdir('InputPictures')

for img in range(0, len(inputDirectory)):
    print(inputDirectory)

    image = cv.imread(inputDirectory[img])

    original = image.copy()

    gray = cv.cvtColor(image, cv.COLOR_BGR2GRAY)

    blur = cv.GaussianBlur(gray, (5,5), 0)

    thresh = cv.threshold(blur, 0, 255, cv.THRESH_BINARY_INV + cv.THRESH_OTSU)[1]

    kernel = cv.getStructuringElement(cv.MORPH_RECT, (3,3))

    dilate = cv.dilate(thresh, kernel, iterations=1)

    dilate_copy = cv.bitwise_not(dilate)

    cnts = cv.findContours(dilate_copy, cv.RETR_EXTERNAL, cv.CHAIN_APPROX_SIMPLE)

    cnts = cnts[0] if len(cnts) == 2 else cnts[1]

    for c in cnts:

        x,y,w,h = cv.boundingRect(c)

        crop_img = thresh[y:y + h, x:x + w]

        height, width = crop_img.shape

        if height > 45 and width > 45:

            cv.imwrite("CroppedPictures\cropped{}.jpg".format(image_number),
crop_img)

            cv.rectangle(image, (x, y), (x + w, y + h), (36,255,12), 2)

            ROI = original[y:y+h, x:x+w]
            image_number += 1
```

Algorithm 4.1 - The process of normalizing the input images

While the pixel values of the images are normalized and saved as cropped images, another crucial step is to ensure that the obtained images have the same orientation as much as possible. To do so, it is required to count the amount of black pixels on each side of the image, which is left, right, up and bottom. The purpose of counting the black pixels is to ensure the minimum rotations

are needed in order to have images facing the same angle as much as possible. This is done by using the sum difference between left and right sides and up bottom sides of the image and this value is kept as reference. The image is rotated 15 degrees and current black pixels and background dimensions are recalculated. If the current amount of pixels is smaller than the sum difference, then the current angle value is memorized. Additionally, it is ensured that the chess piece is kept at the center of the image. Once the rotation changes have been made, the image is saved as well as the values of it as a csv file, as shown in Figure 4.5 and 4.6, on the rook.



Figure 4.5 - Normalized rook image after rotating the image

	A	B	C	D
1	pixelRow;pixelColumn			
2	11.5;0.0			
3	14.166666666666666;0.0			
4	17.38888888888889;0.0			
5	20.462962962962962;0.0			
6	22.820987654320987;0.0			
7	24.940329218106996;0.0			
8	26.31344307270233;0.0			
9	27.104481024234108;0.6666666666666666			
10	27.7014936747447;1.8888888888888886			
11	28.233831224914898;2.9629629629629632			
12	28.744610408304965;3.9876543209876547			
13	28.914870136101655;4.995884773662552			
14	28.971623378700553;5.665294924554185			
15	28.99054112623352;6.2217649748513955			
16	28.663513708744507;9.0739216582838			
17	27.887837902914836;13.024640552761268			
18	26.629279300971614;15.341546850920423			
19	25.543093100323873;17.113848950306807			
20	25.18103103344129;18.704616316768934			
21	25.060343677813762;19.901538772256313			
22	24.68678122593792;20.967179590752107			
23	23.895593741979308;21.989059863584036			
24	21.965197913993105;22.996353287861343			
25	19.655065971331037;25.332117762620445			
26	18.218355323777015;28.777372587540146			
27	17.406118441259007;32.92579086251338			
28	16.80203948041967;38.64193028750446			
<div> <div><></div> <div>distPictures6</div> <div>+</div> </div>				

Figure 4.6 - CSV file of the rook piece

The following algorithm 4.2 shows the process of readjusting the angle of chess pieces:

```
def recalculateBlackPixels(inputImgObjects):

    tmp2 = np.sum(inputImgObjects, axis=2)

    tmp2[tmp2<100] = 0

    bounds = np.where(tmp2==0)

    left, right, top, down = np.min(bounds[0]), np.max(bounds[1]),
    np.min(bounds[1]), np.max(bounds[0])

    if right>down:

        imgObject = inputImgObjects[left:right+1, top:down+1]

    else:

        imgObject = inputImgObjects[top:down + 1, left:right+1]

    height, width, channels = imgObject.shape

    blackPixelCountLeftSide = 0

    blackPixelCountRightSide = 0

    blackPixelCountUpperSection = 0

    blackPixelCountBottomSection = 0

    for column in range(0, height):

        if imgObject[column][width // 2][0] < 10:

            if column < height // 2:

                blackPixelCountLeftSide += 1

            else:

                blackPixelCountRightSide += 1

    for row in range(0, width):

        if imgObject[height // 2][row][0] < 10:

            if row < width // 2:

                blackPixelCountUpperSection += 1

            else:
```



```

        blackPixelCountBottomSection += 1

sumdiff = np.abs(blackPixelCountLeftSide - blackPixelCountRightSide) + \
        np.abs(blackPixelCountUpperSection - blackPixelCountBottomSection)

return sumdiff

for image_index in range(0, image_number):
    inputImgObjects =
    cv.imread("CroppedPictures\cropped{}.jpg".format(image_index))

    height, width, channels = inputImgObjects.shape
    blackPixelCountLeftSide = 0
    blackPixelCountRightSide = 0
    blackPixelCountUpperSection = 0
    blackPixelCountBottomSection = 0
    for column in range(0, height):
        if inputImgObjects[column][width//2][0] < 10:
            if column < height // 2:
                blackPixelCountLeftSide += 1
            else:
                blackPixelCountRightSide += 1

    for row in range(0, width):
        if inputImgObjects[height//2][row][0] < 10:
            if row < width // 2:
                blackPixelCountUpperSection += 1
            else:
                blackPixelCountBottomSection += 1

    rotated = inputImgObjects
    height, width, _ = rotated.shape
    # Calculate the dimensions of the background
    background_height = max(height, width)
    background_width = max(height, width)

```

```

background_color = (255, 255, 255) # White

# Create the background

background = np.zeros((background_height, background_width, 3),
dtype=np.uint8)

background.fill(background_color[0])

# Paste the rotated image onto the background

x = (background_width - width) // 2
y = (background_height - height) // 2
background[y:y + height, x:x + width] = rotated

rotated = background

minPixels = recalculateBlackPixels(rotated)

bestRotation = 0

for angle in np.arange(15, 360, 15):

    center = (rotated.shape[1] // 2, rotated.shape[0] // 2)

    rot_mat = cv2.getRotationMatrix2D(center, angle, 1)

    rotated = cv2.warpAffine(inputImgObjects, rot_mat, (background_height,
background_width), borderValue=(255, 255, 255))

    curPixels = recalculateBlackPixels(rotated)

    if curPixels < minPixels:

        minPixels = curPixels

        bestRotation = angle

    else:

        break

if bestRotation == 0:

    for angle in np.arange(-15, -360, -15): # rotate right

        center = (rotated.shape[1] // 2, rotated.shape[0] // 2)

        rot_mat = cv2.getRotationMatrix2D(center, angle, 1)

        rotated = cv2.warpAffine(inputImgObjects, rot mat.
(background_height, background_width), borderValue=(255, 255, 255))

        curPixels = recalculateBlackPixels(rotated)

        if curPixels <= minPixels:

            minPixels = curPixels

            bestRotation = angle

        else:

            break

```

```

center = (background.shape[1] // 2, background.shape[0] // 2)
rot_mat = cv2.getRotationMatrix2D(center, bestRotation, 1)
height, width, _ = background.shape
# Calculate the dimensions of the background
background_height = max(height, width)
background_width = max(height, width)
bestimg = cv2.warpAffine(background, rot_mat, (background_height,
background_width), borderValue=(255, 255, 255))
cv.imwrite("RotatedPictures\\rotated{}.jpg".format(image_index), bestimg)

```

Algorithm 4.2 - The process of rotating the normalized images

So far, images have been normalized and their angle adjusted. With these changes, their pixel values can be worked with. For easier access and organization of the data between multiple image files, they are stored in a Pandas dataframe. Each column is labeled the images's file name as well as the row and column of the respective image as shown in the algorithm 4.3 below:

```

files = os.listdir('PixelsDistribution')
folderSize = len(os.listdir('PixelsDistribution'))
print (files)
df = None
for file in files:
    tmp = pd.read_csv('PixelsDistribution\\' + file, sep=';')
    cols = tmp.columns
    cols = {x+'_'+file.split('.')[0][12:] for x in cols}
    tmp = tmp.rename(columns={'pixelRow': list(cols)[1], 'pixelColumn':
list(cols)[0]})
    if df is None:
        df = tmp.copy()
    else:
        df[tmp.columns] = tmp.values

```

Algorithm 4.3 - Putting the images into dataframe

Every image's data is stored in the dataframe, which allows for easier computation for the next task, as shown in Figure 4.7. In order to find out which images are similar, a distance measure

can be used to determine how similar the images are. In this case, the Euclidean distance is used to measure.

Figure 4.7 - Dataframe of every image

Euclidean distance can be considered a similarity metric. This metric is used in the popular descriptor SIFT according to Iaroslav Melekhov, Juho Kannala and Esa Rahtu [26]. A distance matrix is used to store the distance computation with each pair of images that are from the dataframe. The computation is done by the distance between any two points on the real line is the absolute value of the numerical difference of their coordinates, their absolute difference [51]. This is shown in the algorithm 4.4 below:

```
distanceMatrix = np.zeros((df.shape[1]//2, df.shape[1]//2))

for i in range(0, df.shape[1]//2):
    for j in range(0, df.shape[1]//2):
        s1 = np.abs((df[f'pixelRow_{i}'] - df[f'pixelRow_{j}']).sum())
        s2 = np.abs((df[f'pixelColumn_{i}'] - df[f'pixelColumn_{j}']).sum())
        distanceMatrix[i,j] = s1+s2
```

Algorithm 4.4 - Creating the distance matrix, using Euclidean distance

The next step is to apply the hierarchical clustering now that we have the distance matrix. The hierarchical clustering is defined in a method that accepts the following parameters: the

distance matrix, minimum objects in a cluster, maximum objects in a cluster and distance threshold. These parameters are responsible for ensuring that an excessive number of clusters are not created as well as not putting too many objects in the cluster and having a minimum required number of objects in order for a new class to be created. This is advantageous for memory usage. The algorithm does not know how many clusters need to be created. In order to help with this, the program makes a comparison of random images within the cluster. The program asks the user to check the two images if they are the same. If the images are different, then the program removes an image. If all the images are faulty, they are all removed which means that they are not similar and the cluster should not exist. This will aid with the creation of new classes for incoming input objects, due to the algorithm being unsupervised.

The algorithm loops through each object in the distance matrix. It condenses the section of the distance matrix that the current object is in and linkage method is used. Condensed distance matrix, must have the same amount of rows and columns. Linkage is used to calculate the distance between two clusters. The algorithm begins with clusters that will be used for forming the hierarchy. Two clusters are combined into a single cluster, the previous two are removed, and the new cluster is created. The algorithm stops, when only one cluster remains. With the final cluster, it becomes the root. The distance matrix must be updated at each iteration, to ensure that the newly formed clusters are up to date and provide accurate measurements for the distance of the newly formed cluster with the rest of the clusters [45].

Next, fcluster method is used to form flat clusters from the hierarchical clustering thanks to the linkage array created. This array represents a dendrogram, where the first and second elements are the two clusters merged at each step, the third element is the distance between these clusters, and the fourth element is the size of the new cluster [44]. In other words, fcluster is useful to find memberships between objects. The unique IDs of the clusters are also stored and each cluster is looped in order to access the objects that belong in those clusters. A variable is used to store indices where the objects are in the respective cluster. This is done by comparing the elements of the object membership with cluster ID. The size of the cluster is also stored in order to check if the size of the cluster is bigger than the minimum number of objects in the cluster parameter. If the condition is met, three random images are taken from the cluster and the user is asked if the two images are the same, as shown in Figure 4.8. There are three cases of image checking:

- 1) in the first case, first and second images are checked
- 2) in the second case, first and last image are checked
- 3) in the third case, the second and last image are checked.



```
Are the images that were shown the same? (y/n)
y
```

Figure 4.8 - The program asks the user if both images are the same

If in the first case, the images are not the same, a different image combination case is used to check. Otherwise, the image will not be used for testing and is deleted from the distance matrix. If all cases are not correct, then all three images are deleted. The reason why the images are deleted is to increase the probability that each cluster contains a single type of object.

As the algorithm continues, the accuracy of matching two images increases. When all images are checked, the results of the hierarchical clustering are shown.

The method of hierarchical clustering is shown in algorithm 4.5:

```
def hierarchicalClustering(distanceMatrix, minObjinCluster, maxObjinCluster,
distanceThreshold):

    df = pd.DataFrame(data=distanceMatrix)

    numberOfObjects = len(distanceMatrix)

    for currentNumberOfObjects in range(minObjinCluster, numberOfObjects):

        if currentNumberOfObjects > df.shape[0]:

            break

        condensed_distance_matrix = df.iloc[:currentNumberOfObjects,
:currentNumberOfObjects].values

        Z = linkage(condensed_distance_matrix, method='single',

                    metric='euclidean')

        objectMembership = fcluster(Z, t=distanceThreshold,
criterion='distance')

        clusterIDs = np.unique(objectMembership)

        for ID in range(0, len(clusterIDs)):

            object_indices = np.where(objectMembership == clusterIDs[ID])[0]
```

```

cluster_size = len(object_indices)

if cluster_size >= minObjinCluster:

    randomImg = np.random.choice(object_indices,3,replace=False)

    imagePath =
"NormalizedPictures\\normalized{}.jpg".format(df.columns[randomImg[0]])

    img1 = cv.imread(imagePath)

    imagePath =
"NormalizedPictures\\normalized{}.jpg".format(df.columns[randomImg[1]])

    img2 = cv.imread(imagePath)


    imageDisplay = np.concatenate((img1, img2), axis=1)

    cv.imshow('Image Comparison', imageDisplay)

    cv.waitKey(0)

    cv.destroyAllWindows()

    inputChoice = None

    while True:

        print("Are the images that were shown the same? (y/n)")

        inputChoice = input()

        if inputChoice == 'y':

            break

        if inputChoice == 'n':

            imagePath =
"NormalizedPictures\\normalized{}.jpg".format(df.columns[randomImg[0]])

            img1 = cv.imread(imagePath)


            imagePath =
"NormalizedPictures\\normalized{}.jpg".format(df.columns[randomImg[2]])

            img2 = cv.imread(imagePath)

```

```

imageDisplay = np.concatenate((img1, img2), axis=1)

cv.imshow('Image Comparison', imageDisplay)

cv.waitKey(0)

cv.destroyAllWindows()

print("Case 1: Are the images that were shown the
same? (y/n)")

inputChoice = input()

if inputChoice == 'y':

    df = df.loc[df.index != df.columns[randomImg[1]]] #
row

    del df[df.columns[randomImg[1]]] # column

    distanceMatrix = df.values

    break

    imagePath =
"NormalizedPictures\\normalized{}.jpg".format(df.columns[randomImg[1]])

    img1 = cv.imread(imagePath)

    imagePath =
"NormalizedPictures\\normalized{}.jpg".format(df.columns[randomImg[2]])

    img2 = cv.imread(imagePath)

    imageDisplay = np.concatenate((img1, img2), axis=1)

    cv.imshow('Image Comparison', imageDisplay)

    cv.waitKey(0)

    cv.destroyAllWindows()

    print("Case 2: Are the images that were shown the
same? (y/n)")

    inputChoice = input()

    if inputChoice == 'y':

```



```

row
df = df.loc[df.index != df.columns[randomImg[0]]] #

del df[df.columns[randomImg[0]]] # column

distanceMatrix = df.values

break

try:

df = df.loc[df.index != df.columns[randomImg[0]]]

del df[df.columns[randomImg[0]]] # column

df = df.loc[df.index != df.columns[randomImg[1]]]

del df[df.columns[randomImg[1]]] # column

df = df.loc[df.index != df.columns[randomImg[2]]]

del df[df.columns[randomImg[2]]] # column

distanceMatrix = df.values

except Exception as e:

print(str(e))

print(df)

print(randomImg)

print(df.shape)

break

distanceMatrix = df.values

print(objectMembership)

print(df.columns)

print({x:y for x,y in zip(df.columns, objectMembership)})

```

Algorithm 4.5 - The custom hierarchical clustering method

5. Results

Once the algorithm finishes running, the results are shown in the console. It prints the object and in which cluster it is. As mentioned before, the input is represented by images with chess pieces in a black background. The images are put in an input folder and it contains ten image files. The total number of chess pieces detected are 43. The value of parameters used are 5 minimum objects in clusters, 20 maximum objects in clusters and the distance threshold's value is 400. Once the algorithm is finished, the image number and clusters where they belong are displayed in the console. The table 5.1 highlights the relationships between the images and clusters:

Img x Cluster	1	2	3	4	5	6	7	8	9	10
0						X				
1										X
2										
3					X					
4										
5	X									
6								X		
7									X	
8					X					
9				X						
10										
11									X	
12		X								
13			X							
14										
15					X					
16										
17										
18		X								
19									X	
20		X								
21										
22	X									
23										
24				X						
25										
26								X		
27		X								
28				X						
29										
30	X									
31										
32										
33									X	
34								X		
35										
36										
37							X			
38	X									
39										
40									X	
41				X						
42									X	
43										

Table 5.1 - Relation between images and clusters

In this table, it can be observed where the detected images belong in which cluster. There are a total of 10 clusters created. Some of the images do not have a cluster assigned because they were deleted. The total count is the following:

- 4 images are in cluster 1
- 4 images are in cluster 2
- 1 image is in cluster 3
- 4 images are in cluster 4
- 3 images are in cluster 5
- 1 image is in cluster 6
- 1 images are in cluster 7
- 3 images are in cluster 8
- 6 images are in cluster 9
- 1 image is in cluster 10

In each cluster, the chess pieces are the following:

- Cluster 1: king, pawn, pawn, knight
- Cluster 2: bishop, knight, king, king
- Cluster 3: pawn
- Cluster 4: pawn, knight, knight, queen
- Cluster 5: pawn, pawn, queen
- Cluster 6: pawn
- Cluster 7: pawn
- Cluster 8: rook, rook, knight
- Cluster 9: knight, king, queen, king, bishop, bishop
- Cluster 10: rook

The accuracy of each cluster is calculated by taking the most common chess piece in the cluster and dividing it by the total number of chess pieces, without taking in consideration the clusters with only one piece. The accuracy of each cluster is the following:

Cluster 1: $(2 / 4) * 100 = 50\%$

Cluster 2: $(2 / 4) * 100 = 50\%$

Cluster 4: $(2 / 4) * 100 = 50\%$

Cluster 5: $(2 / 3) * 100 \sim 66.67 \%$

Cluster 8: $(2 / 3) * 100 \sim 66.67 \%$

Cluster 9: $(2 / 6) * 100 \sim 33.33\%$

The average of all clusters accuracy is:

$$(50\% + 50\% + 50\% + 66.67\% + 66.67\% + 33.33\%) / 6 = 52.77 \%$$

My contributions for this research were the development of the custom hierarchical clustering method, by introducing parameter constraints and using the approach to delete images that were not suitable for being in clusters due not being similar. This method was made to ensure the increase of accuracy in detection of new objects.

The algorithm can be improved in the future by employing better detection methods for more complex backgrounds and objects. Further improvements can be done for the clustering method to have improved accuracy. In the case of image processing, the normalization can be improved to have clearer images for the algorithm to work with.

6. Conclusions

In this research, a program has been developed in Python to identify chess pieces in a black background. It has been used CV2, Matplotlib, Numpy, Os, Pandas, Scipy for the research in order to make it possible. This program is useful for the Computer Vision domain and it helps detect objects in a black background.

It has been shown the process of normalization, which is relevant in the image processing in order to ensure that the program is able to read the images without visual distortions such as chromatic and light variances. The normalization methods that have been used are grayscaling, Gaussian blurring and dilation.

The normalized images have also had their angle adjusted in order to ensure that there are no rotation invariances by calculating the black pixel distribution. With each step of the image processing, the images are saved for easier access.

With all the adjustments done, the images are saved as CSV files. The program will be able to read the pixel values much easier. They are also grouped together in a dataframe and each column of the dataframe represents the pixel row and column of the respective image.

Euclidean distance has been used to calculate distance between pixel values of different images which is useful to see how similar the images are. With this computation, it is stored in a distance matrix.

Hierarchical clustering ensures grouping similar objects into categories and allows the classification of the objects for easier identification. With the extra adjustments done for the hierarchical clustering method, it ensures classification of objects in clusters.

This research has helped in a better understanding of the computer vision domain and learning the processes required for working with the images and ensuring the necessary steps for the detection of the objects present in images.

7. References

- [1] Acharya T, Ray AK. Image processing: principles and applications. John Wiley & Sons; 2005 Oct 3.
https://d1wqtxts1xzle7.cloudfront.net/52877853/image_processing_principles_and_applications_4173-libre.pdf?1493477609=&response-content-disposition=inline%3B+filename%3DImage_processing_principles_and_applicat.pdf&Expires=1720173726&Signature=dhcDjUwf6wrtJwULcbltnXq1JW237mgzHnMC42nNM8p-Uh0aksGgNleITq~Z-1kAj17TFpYK9AdXr3eVqNIlx2fyrJfuS6lnsgr7CvzTx7E0v4ZqKql3lQUZQ3zTkqevgxGnYWP_yqWXAj6WUHVmNyc9UbC3T8CXICdXAVS3Nlbt8ge4PnHtYd49SrLchTpR0xGJWtGNovYj-VpiNcXJfJY4ooNU2jc7zpqh39vzB-~BVn2u6ecNwhRdbZxCL2vZQPoL7ZPf5Yc4LV0qVrD1X3Ortjym3S9xICoABVxsyc8sgyubXVX5hg-h3fNfeISao4barde8GgLLPCMgcXYgOw__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA
- [2] Aziz L, Salam MS, Sheikh UU, Ayub S. Exploring deep learning-based architecture, strategies, applications and current trends in generic object detection: A comprehensive review. Ieee Access. 2020 Sep 3;8:170461-95. <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9186021>
- [3] Castrillón M, Déniz O, Hernández D, Lorenzo J. A comparison of face and facial feature detectors based on the Viola–Jones general object detection framework. Machine Vision and Applications. 2011 May;22:481-94.
https://accedacris.ulpgc.es/bitstream/10553/15075/1/J011_MVA11_preprint.pdf
- [4] Climer S, Zhang W, Joachims T. Rearrangement Clustering: Pitfalls, Remedies, and Applications. Journal of Machine Learning Research. 2006 Jun 1;7(6).
<https://www.jmlr.org/papers/volume7/climer06a/climer06a.pdf>
- [5] Computer image processing and recognition Hall E. Computer image processing and recognition. Elsevier; 1979. https://books.google.ro/books?hl=en&lr=&id=X-EsB7FSIegC&oi=fnd&pg=PP1&dq=image+processing+history&ots=hEGcgpdiDm&sig=0fUYvhFq-wk2hFf5iCOQJw8RdNQ&redir_esc=y#v=onepage&q=image%20processing%20history&f=false
- [6] Dave CP, Joshi R, Srivastava SS. A survey on geometric correction of satellite imagery. International Journal of Computer Applications. 2015 Jan 1;116(12).
https://www.researchgate.net/profile/Rahul-Joshi-18/publication/276128802_A_Survey_on_Geometric_Correction_of_Satellite_Imagery/links/5c6a8a97299bf1e3a5af6807/A-Survey-on-Geometric-Correction-of-Satellite-Imagery.pdf
- [7] Dhruv Parthasarathy. A brief history of CNNs in image segmentation: from R-CNN to Mask R-CNN. Athelas Blog. Apr 22, 2017 <https://blog.athelas.com/a-brief-history-of-cnns-in-image-segmentation-from-r-cnn-to-mask-r-cnn-34ea83205de4>
- [8] Geeks for Geeks. Hierarchical Clustering in Data Mining. 12 Dec, 2023.
<https://www.geeksforgeeks.org/hierarchical-clustering-in-data-mining>
- [9] Gordon AD. Classification. CRC Press; 1999 Jun 17.
https://books.google.ro/books?hl=en&lr=&id=_w5AJtbEz4C&oi=fnd&pg=PP11&dq=clusterin

- g+gordon+(1981)&ots=xxoEM6SbLg&sig=chEUe-Cs_vTAKu8c4ICb_-CUB80&redir_esc=y#v=onepage&q=clustering%20gordon%20(1981)&f=false
- [10] Gordon, A.D., 1987. A review of hierarchical classification. *Journal of the Royal Statistical Society: Series A (General)* 150, 119–137. URL: <https://rss.onlinelibrary.wiley.com/doi/abs/10.2307/2981629>, doi:<https://doi.org/10.2307/2981629>, arXiv:<https://rss.onlinelibrary.wiley.com/doi/pdf/10.2307/2981629>, <https://arxiv.org/pdf/2211.06002>
- [11] Goswami D, Gaur R. Automatic license plate recognition system using histogram graph algorithm. *International Journal on Recent and Innovation Trends in Computing and Communication*. 2014;2(11):3521-7. <https://divtechnosys.com/wp-content/uploads/2021/05/3501-Article-Text-3476-1-10-20180904.pdf>
- [12] Hackeling G. *Mastering Machine Learning with scikit-learn*. Packt Publishing Ltd; 2017 Jul 24.
- [13] HP. Computer History: All About the ENIAC. <https://www.hp.com/ca-en/shop/offer.aspx?p=computer-history-all-about-the-eniac>
- [14] IBM. What is Computer Vision? [Internet]. IBM. <https://www.ibm.com/topics/computer-vision>
- [15] Isahit. Why to use Grayscale Conversion during Image Processing? August 4, 2022 <https://www.isahit.com/blog/why-to-use-grayscale-conversion-during-image-processing>
- [16] J. D. Hunter, "Matplotlib: A 2D Graphics Environment", *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90-95, 2007
- [17] Jain AK, Dubes RC. *Algorithms for clustering data*. Prentice-Hall, Inc.; 1988 Jul 1. https://homepages.inf.ed.ac.uk/rbf/BOOKS/JAIN/Clustering_Jain_Dubes.pdf
- [18] Jain AK, Murty MN, Flynn PJ. Data clustering: a review. *ACM computing surveys (CSUR)*. 1999 Sep 1;31(3):264-323. http://users.eecs.northwestern.edu/~yingliu/datamining_papers/survey.pdf
- [19] JetBrains. PyCharm Features - JetBrains Python IDE. 2000-2024 <https://www.jetbrains.com/pycharm/features/>
- [20] Li J, Lu BL. An adaptive image Euclidean distance. *Pattern Recognition*. 2009 Mar 1;42(3):349-57. https://bcmi.sjtu.edu.cn/~blu/papers/2009/JingLi_Pattern-Recognition_2009.pdf
- [21] Li Z, Liu F, Yang W, Peng S, Zhou J. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*. 2021 Jun 10;33(12):6999-7019. <https://arxiv.org/pdf/2004.02806>
- [22] Lu J, Healy DM. Contrast enhancement via multiscale gradient transformation. In *Proceedings of 1st international conference on image processing 1994 Nov 13 (Vol. 2, pp. 482-486)*. IEEE. <https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=f246bc25799daac8870b0bb7d36f62e234e18c05>
- [23] Mahmoud Harmouch. 17 types of similarity and dissimilarity measures used in data science. *Towards Data Science*. Mar 14, 2021 <https://towardsdatascience.com/17-types-of-similarity-and-dissimilarity-measures-used-in-data-science-3eb914d2681>
- [24] Marino K, Salakhutdinov R, Gupta A. The more you know: Using knowledge graphs for image classification. arXiv preprint arXiv:1612.04844. 2016 Dec 14. <https://arxiv.org/pdf/1612.04844>

- [25] MathWorks. Types of Morphological Operations. 1994-2024 <https://www.mathworks.com/help/images/morphological-dilation-and-erosion.html>
- [26] Melekhov I, Kannala J, Rahtu E. Image patch matching using convolutional descriptors with euclidean distance. In Asian Conference on Computer Vision 2016 Nov 20 (pp. 638-653). Cham: Springer International Publishing. <https://arxiv.org/pdf/1710.11359>
- [27] Mirkin B. Mathematical classification and clustering. Springer Science & Business Media; 2013 Dec 1. <https://link.springer.com/book/10.1007/978-1-4613-0457-9>
- [28] Murtagh F, Contreras P. Algorithms for hierarchical clustering: an overview, II. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 2017 Nov;7(6):e1219. https://eprints.hud.ac.uk/id/eprint/32552/1/DWD_Fmurtagh_31.pdf
- [29] Murtagh F, Contreras P. Algorithms for hierarchical clustering: an overview. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 2012 Jan;2(1):86-97. <https://i2pc.es/coss/Docencia/SignalProcessingReviews/Murtagh2012.pdf>
- [30] Nielsen F, Nielsen F. Hierarchical clustering. Introduction to HPC with MPI for Data Science. 2016:195-211. <https://franknielsen.github.io/Clustering/BookChapter-HierarchicalClustering.pdf>
- [31] Nowak E, Jurie F. Learning visual similarity measures for comparing never seen objects. In 2007 IEEE conference on computer vision and pattern recognition 2007 Jun 17 (pp. 1-8). IEEE. <https://hal.science/hal-00203958/document>
- [32] NumPy. What is NumPy? — NumPy Manual. 2008-2024 <https://numpy.org/doc/stable/user/whatisnumpy.html>
- [33] OpenCV team. About. 2024 <https://opencv.org/about/>
- [34] OpenCV team. Eroding and Dilating. Jul 4 2024 https://docs.opencv.org/3.4/db/df6/tutorial_erosion_dilatation.html
- [35] OpenCV team. Smoothing Images. Jul 4 2024 https://docs.opencv.org/4.x/d4/d13/tutorial_py_filtering.html
- [36] Ozcanli OC, Tamrakar A, Kimia BB, Mundy JL. Augmenting shape with appearance in vehicle category recognition. In 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06) 2006 Jun 17 (Vol. 1, pp. 935-942). IEEE. <https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=c04b4900042da868190807e9c65aa09e87b7fd52>
- [37] Padilla R, Netto SL, Da Silva EA. A survey on performance metrics for object-detection algorithms. In 2020 international conference on systems, signals and image processing (IWSSIP) 2020 Jul 1 (pp. 237-242). IEEE. https://www.researchgate.net/profile/Rafael-Padilla/publication/343194514_A_Survey_on_Performance_Metrics_for_Object-Detection_Algorithms/links/5f1b5a5e45851515ef478268/A-Survey-on-Performance-Metrics-for-Object-Detection-Algorithms.pdf
- [38] Pandas Development Team. pandas documentation. Apr 10, 2024 <https://pandas.pydata.org/docs/>
- [39] Pei SC, Lin CN. Image normalization for pattern recognition. Image and Vision computing. 1995 Dec 1;13(10):711-23. <https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=dd870203a5fe5b28e9c464889b4a0df3af5dc087>
- [40] Phillips D. Image processing in C. Lawrence: R & D Publications; 1994. <https://d1wqtxts1xzle7.cloudfront.net/52921594/cips2ed-libre.pdf?1493716895=&response->

content-

disposition=inline%3B+filename%3DImage_Processing_in_C_Second_Edition.pdf&Expires=1720077365&Signature=JV75rGmstqijuiGhBrD0HJrBWq7WvF9tEsFKkKJW6JM8bbr0CdP~RKAIKSJ1DGh77w8i8jPvf4wSZbRnIYjbvUJ-sQKkIvM8QsRonhNU9hhqQ6iNjMmX3OBZTf76XI0zBq0Zi-Wqk-pmynaVOYOTtjyt5YqXt9SczOrN12G1L8AmZxIsbyz5qTsa4QvLtXq0AIL89trKkJ5sNPbHi395QhXEVJXsSydjsxcH4Q0XaGC1lSaDXZKQNzzUt~WxEkCXkQhQjKm3HS9AAab5-nGZYHvIO-F62DKGQDmlaAvEwx4-56coNu1PF5bdIHUKBfvdpMVZw5An9t~4in8Nxbu2LA__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

[41] Python Software Foundation. os — Miscellaneous operating system interfaces. Jul 03, 2024 <https://docs.python.org/3/library/os.html>

[42] Python W. Python. Python releases for windows. 2021;24. <https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=1f2ee3831eebfc97bfafd514ca2abb7e2c5c86bb>

[43] scikit-learn developers. AgglomerativeClustering. 2007 - 2024 <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html>

[44] SciPy community. fcluster. <https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.fcluster.html>

[45] SciPy community. Linkage. <https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html>.

[46] SuperAnnotate Blog. Convolutional Neural Networks: 1998-2023 Overview. May 15, 2023 <https://www.superannotate.com/blog/guide-to-convolutional-neural-networks>

[47] Szegedy C, Toshev A, Erhan D. Deep neural networks for object detection. Advances in neural information processing systems. 2013;26. <https://proceedings.neurips.cc/paper/2013/file/f7cade80b7cc92b991cf4d2806d6bd78-Paper.pdf>

[48] Van Rossum G, Drake Jr FL. Python tutorial. History. 2010 Sep 28;42(4):1-22. <https://scicomp.ethz.ch/public/manual/Python/3.9.9/tutorial.pdf>

[49] W3Schools.com. Python os Module. 1999-2024 https://www.w3schools.com/python/module_os.asp

[50] Wikimedia Foundation. Digital image processing. Wikipedia. https://en.wikipedia.org/wiki/Digital_image_processing#:~:text=Many%20of%20the%20techniques%20of,facilities%2C%20with%20application%20to%20satellite

[51] Wikimedia Foundation. Euclidean Distance. Wikipedia. https://en.wikipedia.org/wiki/Euclidean_distance

[52] Wikimedia Foundation. Gaussian blur. Wikipedia. https://en.wikipedia.org/wiki/Gaussian_blur

[53] Wikimedia Foundation. Image Registration. Wikipedia. https://en.wikipedia.org/wiki/Image_registration

[54] Wikimedia Foundation. Machine learning. Wikipedia. https://en.wikipedia.org/wiki/Machine_learning

[55] Xu R, Wunsch D. Survey of clustering algorithms. IEEE Transactions on neural networks. 2005 May 9;16(3):645-78. <https://ieeexplore.ieee.org/document/1427769>