**Connect the Color Dot’s An IQ Test Puzzle Game**

|  |
| --- |
|  |

**By:**

**Muhammad Nofal Ullah**

**29409**

**Sulman Siddique**

**22816**

**Uzair Asif  
22664**

**Supervised by:**

**Professor Muhammad Mansoor Alam**

**Syed Muhammad Waqar Ali**

**Faculty of Computing**

**Riphah International University, Islamabad**

**Fall 2024**

**A Dissertation Submitted To**

**Faculty of Computing,**

**Riphah International University, Islamabad**

**As a Partial Fulfillment of the Requirement for the Award of the Degree of**

**Bachelors of Science in Computer Science**

**Faculty of Computing**

**Riphah International University, Islamabad**

Date: date of final presentation

**Final Approval**

This is to certify that we have read the report submitted by **Muhammad Nofal Ullah**

**(29409)** **Sulman Siddique**(**22816) Uzair Asif (22664)** for the partial fulfillment of the requirements for the degree of the Bachelors of Science in Computer Science (BSCS). It is our judgment that this report is of sufficient standard to warrant its acceptance by Riphah International University, Islamabad for the degree of Bachelors of Science in Computer Science (BSCS).

**Committee:**

|  |  |
| --- | --- |
| **1** | Professor M. Mansoor Alam  Syed Muhammad Waqar Ali  (Supervisor) |
|  |  |
| **2** | Dr. Musharaf  (Head of Department/chairman) |

**Declaration**

We hereby declare that this document “**Connect the Color Dot’s An IQ Test Puzzle game**” neither as a whole nor as a part has been copied out from any source. It is further declared that we have done this project with the accompanied report entirely on the basis of our personal efforts, under the proficient guidance of our teachers, especially our supervisor **M. Mansoor Alam and Syed Muhammad Waqar Ali**. If any part of the system is proved to be copied out from any source or found to be reproduction of any project from anywhere else, we shall stand by the consequences.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**[Muhammad Nofal Ullah** **]**

**[29409]**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**[Sulman Siddique]**

**[22816]**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**[Uzair Asif]**

**[22664]**

**Dedication**

I dedicate this project, Connect the Color Dots IQ Test Puzzle Game, to my beloved family and friends, whose constant encouragement and unwavering support have been my source of strength throughout this journey. A special thanks to my teachers and mentors for their guidance, knowledge, and inspiration, which have shaped my academic endeavors. This project is also dedicated to all those who strive for creativity, innovation, and knowledge in the field of game development and cognitive research.**Acknowledgement**

First of all we are obliged to Allah Almighty the Merciful, the Beneficent and the source of all Knowledge, for granting us the courage and knowledge to complete this Project.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**[Hashim Shahzad]**

**[Friend]**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**[Name of Student 2]**

**[CMS #]**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**[Name of Student 3]**

**[CMS #]**

Table of Contents

[Table of Contents i](#_Toc113957278)

[List of Tables iii](#_Toc113957279)

[List of Figures iv](#_Toc113957280)

[Abstract **Error! Bookmark not defined.**](#_Toc113957281)

[Chapter 1: Introduction 2](#_Toc113957282)

[1.1 Goals and Objectives 3](#_Toc113957283)

[1.2 Scope of the Project 4](#_Toc113957284)

[Chapter 2: Literature Review 5](#_Toc113957285)

[2.1 Introduction 5](#_Toc113957286)

[2.2 Background and Problem Elaboration 5](#_Toc113957287)

[2.3 Detailed Literature Review 7](#_Toc113957288)

[2.3.1 Definitions 7](#_Toc113957289)

[2.3.2 Related Research Work 1 7](#_Toc113957290)

[2.3.3 Related Research Work 2 9](#_Toc113957291)

[2.3.4 Related Research Work 3 10](#_Toc113957291)

[2.4 Literature Review Summary Table 12](#_Toc113957292)

[2.5 Research Gap 13](#_Toc113957293)

[2.6 Problem Statement 14](#_Toc113957294)

[2.7 Conclusion 15](#_Toc113957294)

[Chapter 3: Requirements and Design 16](#_Toc113957295)

[3.1 Requirements 17](#_Toc113957296)

[3.1.1 Functional Requirements 17](#_Toc113957297)

[3.1.2 Non-Functional Requirements 17](#_Toc113957298)

[3.1.3 Hardware and Software Requirements 18](#_Toc113957299)

[3.2 Proposed Methodology 21](#_Toc113957300)

[3.3 System Architecture 23](#_Toc113957301)

[3.4 Use Cases 24](#_Toc113957302)

[3.4.1 Sample Use Case Name Here **Error! Bookmark not defined.**](#_Toc113957303)

[3.5 Database Design *(Optional)* **Error! Bookmark not defined.**](#_Toc113957304)

[3.6 Class diagram *(Optional)* **Error! Bookmark not defined.**](#_Toc113957305)

[3.7 Sequence diagram *(Optional)* **Error! Bookmark not defined.**](#_Toc113957306)

[3.8 Any Other Artifact… **Error! Bookmark not defined.**](#_Toc113957307)

[3.9 Graphical User Interfaces(GUI) (*Optional)* **Error! Bookmark not defined.**](#_Toc113957308)

[Chapter 4: Implementation and Test Cases 27](#_Toc113957309)

[4.1 Implementation 27](#_Toc113957310)

[4.1.1 Implementation of First Component/Algorithm 27](#_Toc113957311)

[4.2 Test case Design and description 27](#_Toc113957312)

[4.2.1 Sample Test case No.1 27](#_Toc113957313)

[4.2.2 Sample Test case No.2 27](#_Toc113957314)

[4.3 Test Metrics 28](#_Toc113957315)

[4.3.1 Sample Test case Matric.No.1 28](#_Toc113957316)

[4.3.2 Sample Test case Metric.No.2 28](#_Toc113957317)

[4.3.3 Sample Test case Metric.No.3 28](#_Toc113957318)

[Chapter 5: Experimental Results and Analysis 29](#_Toc113957319)

[Chapter 6: Conclusion and Future Directions 30](#_Toc113957320)

[References 31](#_Toc113957321)

[Appendix 32](#_Toc113957322)

[Appendix A: Guidelines 32](#_Toc113957323)

[Appendix B: Heading of Sample Appendix B 32](#_Toc113957324)

[Formatting Guidelines 33](#_Toc113957325)

[Chapter 1: Heading 1 33](#_Toc113957326)

[1.1 Heading 2 33](#_Toc113957327)

[1.1.1 Heading 3 33](#_Toc113957328)

[Tables and Figures 34](#_Toc113957329)

[Equations 35](#_Toc113957330)

[Header/Footer 35](#_Toc113957331)

[Other Formatting Guidelines 35](#_Toc113957332)

[References 35](#_Toc113957333)

List of Tables

[Table 1: This is Sample table caption 34](#_Toc55559466)

[Table 2: This is Sample table caption 34](#_Toc55559467)

List of Figures

Figure 1: List of Styles 34

Figure 2: IEEE Reference style 35

**Abstract**

The "Connect the Color Dots IQ Test Puzzle Game" is designed as an interactive and engaging puzzle game aimed at enhancing cognitive abilities through problem-solving. This project integrates the classic dot-connecting puzzle concept with a modern twist of color association to test and improve a player's IQ. The game presents players with a grid of colored dots, and the challenge is to connect matching colored dots without overlapping the connecting lines. The puzzle becomes increasingly difficult as the grid size expands and the number of colored dots increases.The primary objective of this project is to develop a user-friendly, visually appealing puzzle game that challenges the logical thinking and spatial reasoning of players. It also seeks to explore the relationship between problem-solving in puzzle games and cognitive improvement, making it not only entertaining but also educational. The game is designed to be adaptable to different skill levels, offering a dynamic experience for both beginners and advanced players.This project is developed using C sharp and its gaming libraries, ensuring a smooth and responsive user interface. The game mechanics and design considerations are discussed in detail, with emphasis on user engagement and cognitive stimulation.

**Chapter 1**

**Introduction**

**Chapter 1**

# In recent years, puzzle games have evolved from simple pastimes into sophisticated tools for cognitive development[1]. As digital gaming continues to expand, its impact on cognitive skills such as memory, logical reasoning, and spatial awareness has become increasingly apparent. “Connect the Color Dots” is designed not only as an engaging and enjoyable experience but also as a platform for players to assess and enhance their IQ levels[2]. By connecting dots of the same color without overlapping, players must devise strategies that require logical reasoning and spatial awareness. This game seeks to merge entertainment with cognitive enhancement, aiming to provide a mental workout that adapts to each player’s abilities[3]. The introduction of two primary modes, Practice Mode and IQ Mode, caters to a range of skill levels and goals, from casual play to IQ assessment. Additionally, AI-generated levels bring an element of novelty to each session, ensuring that players face diverse challenges that develop their problem-solving skills in new and unexpected ways[4]. The personalized scoring system, which factors in age, time, and moves taken, allows for a dynamic and meaningful IQ measurement that resonates with each player’s unique profile. This project reflects a broader trend in the gaming industry, where games are being recognized not only for their entertainment value but also for their potential to serve as tools for learning and self-assessment.

# 1.1 Introduction

The digital gaming industry has seen tremendous growth in recent years, with puzzle-based games being a key contributor to this success[5]. Puzzle games not only entertain but also challenge a player's mental faculties, improving their problem-solving skills and cognitive abilities[6]. In this project, we introduce “Connect the Color Dots,” an IQ test puzzle game that goes beyond simple entertainment, offering players a way to assess and improve their IQ levels. The game’s main premise revolves around connecting color dots on a grid without overlapping paths. It is designed to engage players by providing challenges that scale in difficulty as they progress through the levels.

This project is being developed as part of a final year academic endeavor. It combines game development, cognitive science, and artificial intelligence to deliver a comprehensive puzzle game experience. The game is particularly focused on testing the user’s IQ through strategically designed levels that involve critical thinking, logic, and spatial reasoning.

The game is divided into two primary modes: Practice Mode and IQ Mode. In Practice Mode, players can freely explore puzzles without any time constraints, allowing them to become familiar with the mechanics. In IQ Mode, puzzles are timed, and a player’s performance is scored to provide an estimate of their IQ. The scoring system incorporates age, moves taken, levels completed, and time spent to give a holistic assessment of the player’s cognitive abilities.

The levels in this game are generated using artificial intelligence (AI), ensuring that each level presents a unique challenge. This feature not only enhances the replayability of the game but also ensures that the player’s IQ is tested against a variety of scenarios. In addition, the project incorporates a login system where the player’s information, including age and performance data, is stored. This allows the game to tailor its challenges to the player’s profile, making the IQ assessment more personalized and accurate.

This chapter will focus on outlining the goals, objectives, and scope of the project to provide a clear understanding of the direction and purpose of the game.

**1.2 Goals and Objectives**

The primary goal of the "Connect the Color Dots" project is to develop an interactive, engaging puzzle game that tests and evaluates a player's cognitive abilities through IQ-based scoring[7]. Unlike traditional puzzle games, this project integrates elements of cognitive science to ensure that the game provides accurate assessments of a player's IQ. By the end of the project, the game will not only serve as a form of entertainment but also as an educational tool for cognitive improvement.

**1.2.1 Specific Objectives:**

#### ****1.2.1.1 Develop a Puzzle Game with Intuitive Mechanics****:

#### The core gameplay involves connecting matching colored dots on a grid. The objective is to create a user-friendly interface that is easy to navigate, allowing players to understand the game mechanics quickly. Players will be tasked with solving increasingly complex puzzles, requiring them to use logic and critical thinking[8].

#### 1.2.1.2 ****Implement IQ Testing Based on Performance****:

The game's IQ Mode will calculate the player's IQ based on several factors, including age, the number of moves taken to complete a puzzle, the time taken to solve each level, and the number of levels completed. This scoring mechanism will provide a reliable assessment of the player’s cognitive performance.

**1.2.1.3 Use Artificial Intelligence for Level Generation:**

The game will use AI algorithms to generate unique levels, ensuring that no two levels are the same. This feature will enhance the replayability of the game, providing new challenges every time the player starts a new game. AI-based level generation also ensures that the game scales in difficulty based on the player’s progress[9].

**1.2.1.4 Develop Two Gameplay Modes: Practice and IQ Mode:**

In Practice Mode, players will be able to play without any time or move constraints, allowing them to explore and learn at their own pace. In IQ Mode, players will face a timed challenge, with their performance directly influencing their IQ score. This dual-mode gameplay ensures that the game caters to both casual players and those interested in testing their IQ.

**1.2.1.5** **Create a Personalized User Experience:**

The project includes a login system where users must input their age and username. The game will store this information and adjust the difficulty and scoring based on the player's profile. The player's performance data, including completed levels and IQ scores, will be saved, allowing them to track their progress over time.

**1.2.1.6 Design an Educational Tool for Cognitive Improvement:**

While the game is primarily entertainment-based, it also aims to educate players by improving their logical reasoning and problem-solving skills. Through repeated gameplay, players can track their cognitive improvement, making the game not only fun but also beneficial for mental development.

## 1.2 Scope of the Project

The project scope includes the following key components:

* Creation of own custom dataset.
* Development of preprocessing techniques for standardization, noise reduction,

and orientation correction of input images.

* Implementation of feature extraction methods for identifying key UML Class Diagram symbols and relationships.
* Integration of OCR technology for textual recognition to enhance the completeness of the digital representation.
* Design and implementation of a robust model architecture, utilizing machine learning techniques for accurate symbol recognition.
* Validation of the solution to ensure effectiveness across various styles and complexities of hand-drawn UML Class diagrams.
* Presentation of the output in a structured XML format, encapsulating both recognized symbols and textual annotations for seamless integration into existing UML tools.

# Chapter 2:

# Literature Review

## 2.1 Introduction

We will conduct a thorough analysis of relevant literature in this chapter, offering insights into the difficulties involved in transforming hand-drawn UML diagrams into digital formats. We will examine current approaches, their drawbacks, and the noted area of unmet research need. This chapter will also outline our project's history and problem statement, laying the groundwork for the creation of software that can recognize hand-sketched UML diagrams and transform them into editable digital representations. We hope that this review will provide some background and motivation for our work while emphasizing the need for an automated solution to improve workflow, efficiency, and collaboration in work environments.

## 2.2 Background and Problem Elaboration

During meetings or brainstorming sessions, people frequently turn to hand sketching their ideas and concepts in professional settings. One of the biggest challenges is converting these handdrawn sketches especially UML (Unified Modeling Language) diagrams—into digital formats. In fast-paced work environments, efficiency and collaboration are hampered by this labor- and time-intensive manual conversion process.

In the early stages of a project, UML models are usually hand-sketched; however, these sketches cannot be directly imported into UML tools that are currently in use. The inability of current modeling tools, like draw.io, to support image imports makes it challenging to automatically transform hand-drawn sketches into digital formats. The inability of UML CASE tools to extract UML models from images results in problems with interoperability between different tools and formats, even though they provide a variety of features for creating, editing, and exporting UML models, including exporting them as images.

Because of this, the process of manually converting hand-drawn UML diagrams into digital formats is still laborious and ineffective, which interferes with professional workflow and creates obstacles to collaboration. Our project intends to close this gap by creating a system that recognizes and converts activity diagrams and class diagrams automatically. This will make it simple for stakeholders to incorporate their hand-drawn sketches into current digital platforms for additional editing and sharing.

Professionals in a variety of industries who regularly generate and communicate ideas through hand-drawn sketches will greatly benefit from this advancement, as will software developers and engineers who must digitize hand-drawn UML diagrams for further development and documentation.

## 2.3 Detailed Literature Review

We offer a thorough analysis of the literature in this section that is pertinent to our project, which is to automatically extract UML Class and Activity diagrams from hand-drawn sketches. After providing definitions for important terms and concepts associated with our study, we critically examine previous studies conducted in the area.

### 2.3.1 Definitions

Before delving into the literature review, it's important to define key terms used in our project:

* UML (Unified Modeling Language): A standardized modeling language used in software engineering for visualizing, specifying, constructing, and documenting the artifacts of a software system.
* UML Class Diagram: A type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations, and the relationships among objects.
* UML Activity Diagram: A type of behavior diagram that depicts the dynamic aspects of a system by modeling the flow of control from activity to activity.

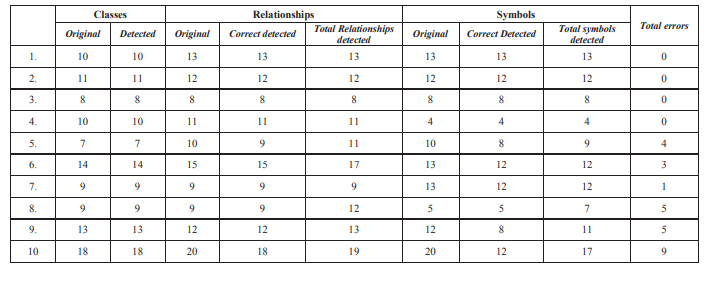
### 2.3.2 Related Research Work 1

Several research works have explored the automatic extraction of UML diagrams from hand-drawn sketches. One notable study is "Extracting UML models from images" by Rodrigues et al. (2013). This research presents the Img2UML tool, which uses image processing techniques to recognize classes, relationships, and symbols from hand-drawn UML diagrams. The tool aims to address the challenge of updating UML models that are stored in image formats, making them difficult to modify. Rodrigues et al. demonstrate the functionality of Img2UM and validate its accuracy in detecting classes, relationships, and symbols.

**2.1 Strengths:**

Img2UML tackles a prevalent issue in software engineering by offering a workable solution for extracting UML Class models from images.

As can be seen in the table, the tool's validation results show how well it can identify classes, relationships, and symbols, suggesting its potential utility.

  
 **Table 2.1: Validations results of IMg2UML tool**

**2.2 Limitations:**

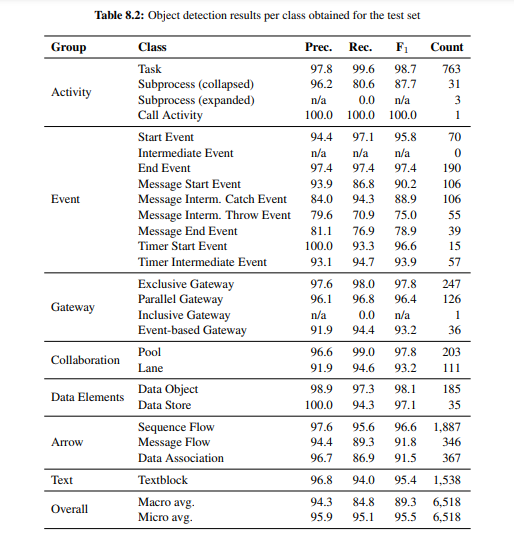
The tool's performance can vary based on the complexity and quality of the input images, especially in recognizing symbols and text. Moreover, it's essential to note that the tool doesn't recognize hand-drawn sketches rather, it focuses on the recognition and conversion digital images. Additionally, it's important to highlight that the tool does not recognize all the elements of a class diagram; instead, it covers only a subset of the elements typically found in class diagrams. Furthermore, while the tool addresses the recognition and transformation of UML Class Diagrams, it does not explore the extraction of UML Activity diagrams, which are also vital in software design.

### 2.3.3 Related Research Work 2

Sketch2Process is a tool that automates the recognition of hand-drawn diagrams, with a specific focus on business process diagrams created on paper. It introduces the hdBPMN dataset, comprising 702 business process diagrams collected from 107 participants, along with BPMN annotations, made publicly available to facilitate further research. Addressing the challenges inherent in recognizing diagrams drawn on paper, Sketch2Process uses neural network-based techniques to detects shapes, arrows, and textblocks, while also recognizing edges and labels, ultimately generating BPMN XML files suitable for process modeling tools. Overall, Sketch2Process contributes valuable insights and methodologies to automated diagram recognition, offering practical solutions to longstanding challenges.

**2.3.3.1 Strengths:**

Sketch2Process shows its strong strengths in detection and recognition of various BPMN elements as shown in table 2.2. It provides a strong solution for recognition of BPMN models and also provides an environment where these models can be edited in a real time after recognition and extraction from images and generates a BPMN XML files which can be imported to modeling tools for further editing and utilization.



**Table 2.2: Results of Sketch2Process Object detection results per class.**

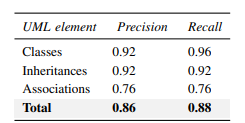
**2.3.3.2 Limitations:**

Sketch2Process provides a strong solution for automating the process of BPMN models but with a partial capability to recognize activity diagrams as most of the elements of activity diagrams are similar to flow charts. However, it lacks the ability to recognize class diagrams.

### 2.3.4 Related Research Work 3

### SketchToPlantUML is a pioneering tool designed to bridge the gap between informal hand-drawn UML Class Diagrams and formal PlantUML models, a crucial step in software engineering. By harnessing the power of the OpenCV library, this tool automates the transformation process, significantly reducing the time and effort required for manual conversion. It preprocesses images, segments UML elements, identifies geometric features, and classifies relationships, culminating in the generation of equivalent PlantUML models.

This tool addresses the inherent challenge of manually transforming transient sketches into reusable and shareable formal models. Its effectiveness is underscored by rigorous testing against a dataset of 70 sketched UML Class Diagrams, where it achieved impressive Precision and Recall values of 88% and 86% respectively. Notably, it excels in accurately recognizing classes, with scores of 0.92 for Precision and 0.96 for Recall, while also demonstrating competence in handling association relationships.



**Tale 2.3: Precision and Recall Evaluation Metrics for UML Element Classification.**

**2.3.4.1 Strengths:**

Sketch2Plant UML shows its high accuracy in recognition of UML elements such as classes, inheritance, associations and conversion to formal PlantUML models as shown in figure 2.3.

**2.3.4.2 Limitations:**

SketchToPlantUML excels in accurately recognizing hand-sketched UML Class Diagrams, focusing primarily on identifying classes, inheritance, and association relationships. However, its scope is limited to class diagrams and also shows its limitations in recognizing text from sketches. Additionally, while the tool simplifies the transformation process, users need prior knowledge of PlantUML coding conventions to fully utilize its capabilities.

## 2.4 Literature Review Summary Table

The columns in the table depend upon your problem and should be specific to your project.

**Table 1: History of Hand-Drawn Diagram Recognition and Conversion Methods**

The summary of various methods and tools developed for recognizing and converting hand-drawn diagrams into digital formats

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No.** | **Name, reference** | **Inventor** | **Year** | **Input** | **Output** | **Description** |
| 1. | [Sketch 2Process](https://madoc.bib.uni-mannheim.de/64778/2/doctoral_thesis.pdf) | B. Schäfer et al. | 2023 | Hand- drawn Process Models | Digital BPMN Models | Automates the conversion of hand-drawn process models into digital BPMN models using neural network-based techniques. |
| 2. | [Img2UML](https://www.researchgate.net/publication/261493373_Extracting_UML_models_from_images?enrichId=rgreq-55d08642-3dbb-4416-8462-0fed5f9aad29&enrichSource=Y292ZXJQYWdlOzI2MTQ5MzM3MztBUzoyNDA5MTAyODU5MzA1MDBAMTQzNDQ0ODg5NjY3NA%3D%3D&el=1_x_2) | Michel Chaudron,  B. Karasneh | 2013 | Digital  Images of UML  diagrams | Extracted UML Models  From Images | Aims to extract UML models from images, assisting in the digitization of UML diagrams for further editing and use |
| 3. | [Sketch](https://miun.diva-portal.org/smash/record.jsf?faces-redirect=true&aq2=%5B%5B%5D%5D&af=%5B%5D&searchType=SIMPLE&sortOrder2=title_sort_asc&query=&language=en&pid=diva2%3A1786365&aq=%5B%5B%5D%5D&sf=all&aqe=%5B%5D&sortOrder=author_sort_asc&onlyFullText=false&noOfRows=50&dswid=-2503)  [2PlantUML](https://miun.diva-portal.org/smash/record.jsf?faces-redirect=true&aq2=%5B%5B%5D%5D&af=%5B%5D&searchType=SIMPLE&sortOrder2=title_sort_asc&query=&language=en&pid=diva2%3A1786365&aq=%5B%5B%5D%5D&sf=all&aqe=%5B%5D&sortOrder=author_sort_asc&onlyFullText=false&noOfRows=50&dswid=-2503) | Monique  Axt | 2023 | Hand Sketch of Class Diagram | Formal PlantUML  Model. | Automates the transformation of hand-drawn UML Class Diagrams into formal PlantUML models using the OpenCV library. |

## 2.5 Research Gap

In the field of UML diagram recognition and conversion, several tools have emerged, each addressing specific needs but also presenting significant limitations. Img2UML, for instance, extracts UML Class models from image files and converts them into XMI files compatible with CASE tools like StarUML. However, it fails to recognize hand-drawn sketches, which limits its applicability in scenarios where such sketches are common.

On the other hand, Sketch2PlantUML focuses on transforming hand-drawn UML Class Diagrams into formal PlantUML models using OpenCV for image processing. Despite its automation capabilities, its output can only be viewed with the PlantUML tool, requiring users to have knowledge of PlantUML coding for any modifications.

SketchUML offers a natural, paper-like interface for students to create UML class diagrams directly within a digital environment, enhancing the learning experience. However, it does not support converting hand-drawn sketches into an editable digital format.

Lastly, Sketch2Process automates the recognition of hand-drawn business process diagrams, generating BPMN XML files for use in process modeling tools. While effective for business process models, it only partially recognizes activity diagrams and does not support class diagrams.

The primary research gap lies in the absence of a comprehensive system capable of recognizing and converting various types of diagrams, such as activity and class diagrams, into formats that can be seamlessly exported to popular modeling tools. Existing tools either focus on specific diagram types or lack integration with multiple modeling tools. For example, Img2UML offers compatibility with tools like StarUML but does not support hand-drawn diagrams. Similarly, Sketch2Process excels at business process models but falls short on class diagram recognition.

Therefore, there is a significant need for a versatile and efficient system that can handle multiple diagram types and provide direct integration with popular modeling environments.

## 2.6 Problem Statement

People commonly use hand sketching in business contexts to communicate ideas and concepts, especially during brainstorming or meeting sessions. However, there is a big obstacle in translating these hand-drawn sketches especially UML (Unified Modeling Language) diagrams into digital formats. The labor-and time-intensive nature of the manual conversion process hinders productivity and teamwork in hectic work settings. Inefficiencies like time consumption, loss of detail, obstacles to collaboration, and disruption of workflow result from this lack of an automated solution.

One of the most commonly used types of UML diagrams in professional settings are class diagrams and activity diagrams. Despite their importance, there is a noticeable gap in automated tools capable of recognizing hand-drawn UML diagrams and converting them into editable digital representations. Existing solutions either focus on specific diagram types, lack compatibility with popular modeling tools, or are limited in functionality.

A flexible and effective system that can manage various diagram types and offer direct integration with well-known modeling environments is therefore desperately needed. We are proactively addressing this gap by creating a solution that recognizes and converts class diagrams and activity diagrams automatically. Although our system might not be able to address every issue that currently exists, it is a significant step in the right direction toward a comprehensive tool that improves accessibility and usability for users in a variety of contexts.

**2.7 Conclusion:**

This chapter outlined the project's background and problem statement, as well as offered a thorough analysis of pertinent literature. The literature review addressed the difficulties in transferring hand-drawn UML diagrams into digital formats as well as the shortcomings of current solutions. This review revealed that, although certain types of diagrams have tools available, or tools with limited functionality, there is a glaring gap in the creation of a comprehensive system that can identify and translate various UML diagram types into editable digital representations.

Our project aims to address this gap by developing a software system that can recognize class diagrams and activity diagrams, and convert them into editable digital formats. By automating this process, our system will enable professionals to seamlessly integrate their hand-drawn sketches into existing digital platforms, improving collaboration, saving time, and enhancing workflow efficiency.

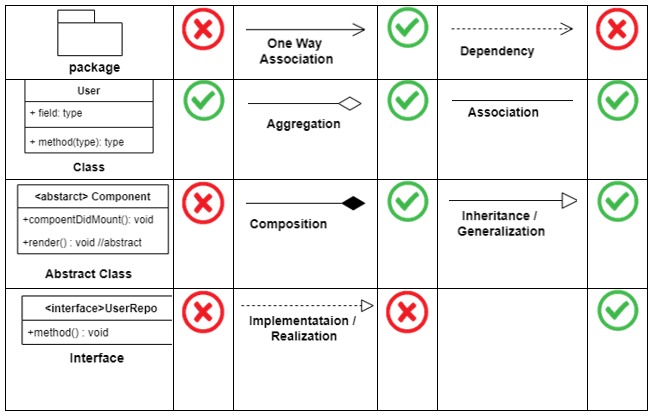
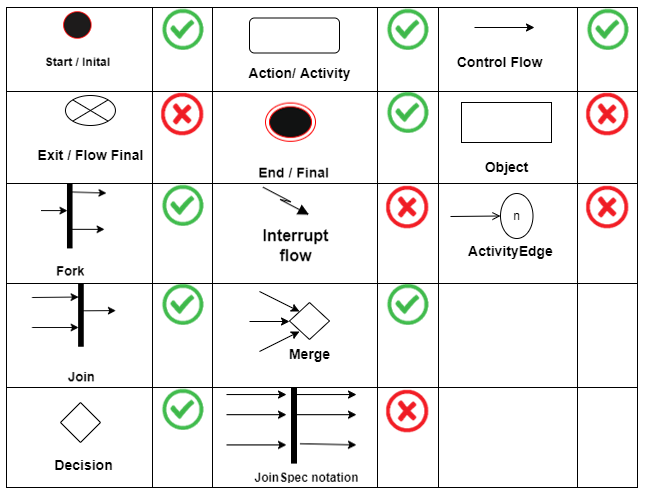
We have defined the background and problem statement, carried out an extensive review of relevant literature, and laid the groundwork for our project in this chapter. Continuing from the insights gathered from the literature review, Chapter 3 will concentrate on detailing the methodology and approach we will use to develop our solution.

**Chapter 3:**

**Requirements And Design**

The design and methodology of our project, which recognizes and converts hand-drawn UML diagrams into digital formats automatically, are covered in Chapter 3. It covers prerequisites for software and hardware, necessary functional and non-functional requirements, and the particular libraries and frameworks that are used. The process of creating and annotating a custom dataset, developing preprocessing, feature extraction, and OCR integration algorithms, and training and assessing these algorithms are all covered in the methodology section. The system architecture is shown, including the user interface and recognition engine details, as well as the parts that go into identifying UML elements and creating XMI files. A use case discussion on creating XMI files from hand-drawn UML diagrams closes the chapter.

Our project focuses on automating the recognition and conversion of UML diagrams, specifically targeting the most widely used notations for class diagrams and activity diagrams as specified by the OMG. To achieve this, we have identified and incorporated the most widely used notations in the creation of these diagrams. Figure 3.1 illustrates the activity diagram notations addressed by our project, while Figure 3.2 details the class diagram notations encompassed. By automating these essential notations, we lay a solid foundation for the requirements and design phase, ensuring our system effectively meets the needs of software development processes.

**Table 3.1: Activity Diagram Symbol Table 3.2: Class Diagram Symbol**

## 3.1 Requirements

### 3.1.1 Functional Requirements

|  |  |
| --- | --- |
| ID | Requirements |
| FR - 1.1 | Users should be able to upload hand-drawn UML diagrams in Jpg and PNG format |
| FR - 1.2 | The system should accurately identify and recognize UML elements such as classes, attributes, methods, relationships, and activities from the uploaded images. |
| FR – 1.3 | Upon recognition of UML elements, the system should generate a Star UML-compatible XMI file representing the digital version of the hand-drawn UML diagram. |

### 3.1.2 Non-Functional Requirements

Sure, here are the rewritten non-functional requirements:

* **Accuracy**

The system must recognize hand-drawn UML diagrams to ensure dependable conversion into digital representations.

* **Usability:**

The user interface shall be intuitive and user-friendly, allowing users to easily upload hand-drawn diagrams and download the generated XML files.

* **Compatibility:**

The output XML format should be compatible with existing UML tool and standards, ensuring smooth incorporation into software development workflows.

* **Documentation:**

Comprehensive documentation, including user guides and technical specifications, shall be provided to assist users and developers in understanding and using the system effectively. User guides must clearly state that more complex diagrams or diagrams larger than standard normal sized papers may result in failure of XMI file generation.

### Hardware and Software Requirements

For training the model on the dataset, the hardware requirements can vary depending on the size of the dataset, the complexity of the model architecture, and the training approach. However, for moderate sized datasets following requirements are considered Adequate.

* **GPU (Graphics Processing Unit):**

A dedicated GPU will be used for accelerating the training process, especially for deep learning models. GPUs with CUDA support from NVIDIA are commonly used for training neural networks due to their parallel processing capabilities.

* **CPU (Central Processing Unit):**

A multi-core CPU is required to handle preprocessing tasks, manage data flow, and coordinate training processes. A modern CPU with at least 4 cores and a high clock speed is recommended. Models such as Intel Core i7 or AMD Ryzen 7 are suitable for training deep learning models.

* **RAM (Random Access Memory):**

Sufficient RAM is necessary to hold the model parameters, intermediate computations, and data during training.

A minimum of 16 GB of RAM is recommended for training moderate-sized models.

* **Storage:**

Adequate storage space is needed to store the dataset, model checkpoints, and training logs.

A high-speed SSD (Solid State Drive) with sufficient capacity (at least 500 GB) is recommended for storing datasets and model files.

* **Power Supply:**

A stable power supply is crucial to ensure uninterrupted training sessions, especially for GPU-intensive tasks.

**3.1.3 Software Requirements:**

* **Operating System**:

The operating system is required for making our system

* **Roboflow.:**

The roboflow platform will be used for custom dataset labelling and annotating.

* **Pycharm:**

This platform will be used for writing python code.

* **Programming Languages:**

The primary language for developing the software system is likely Python, given its popularity in machine learning and image processing domains.

* + - 1. **Libraries and Frameworks:**
* **OpenCV**:

Computer vision library will be used for image preprocessing, feature extraction to standardize, reduce noise, and correct the orientation of hand-drawn UML diagrams.

* **Model:**

Deep learning object detection model will be trained on the custom dataset for detecting UML elements in image.

* **TensorFlow or PyTorch:**

Deep Learning frameworks will be used for constructing and training complex neural network models for recognizing UML symbols and relationships in images.

* **Tesseract OCR:**

OCR will be used for recognizing textual annotations within the diagrams due to its flexibility and ease of integration with python.

* **XML Processing:**

XML Processing Libraries like xml.etree.ElementTree in Python for parsing and generating XML files, ensuring compatibility with existing UML tool such as StarUML.

## Proposed Methodology

**3.2.1 Dataset:**

One of the most significant challenges we face is the absence of available datasets. Currently, there are no datasets specifically tailored to hand-sketched activity diagrams and class diagrams. This absence of relevant datasets has hindered progress in developing recognition systems for these types of diagrams. To address this gap, we are taking steps to create a new dataset explicitly designed for recognizing hand-sketched activity and class diagrams.

**3.2.2 Dataset Labelling and Annotation:**

The next big challenge to tackle is the annotation of the dataset. As, the dataset creation of was custom so we had to annotate the images because the deep learning-based models required the dataset into a labelled format which contains images folder, and a label’s folder, so the we had to manually annotate the entire dataset which consisted of drawing bounding boxes and assigning a label to each images based on the behavior of each diagram element.

#### 3.2.3 Development of Algorithms:

* **Preprocessing Methods:**

Develop techniques for noise reduction, orientation correction, and image standardization to enhance the quality and consistency of input images.

* **Feature Extraction:**

Design algorithms to extract important UML relationships and symbols from preprocessed images, distinguishing between various UML elements.

* **OCR Integration:**

Integrate OCR technology to recognize textual annotations within the diagrams, ensuring accurate extraction of textual information.

* **Model Architecture:**

Design and implement a robust model architecture, potentially utilizing deep learning methods, for precise symbol identification. Experiment with different architectures and configurations to optimize performance.

#### 3.2.4 Training and Evaluation:

* **Training Process:**

Train the developed algorithms and model architectures using the annotated dataset, ensuring diverse representations of different symbol types.

* **Validation:**

Split the dataset into training, validation, and test sets to evaluate the performance of the trained models. Employ cross-validation techniques to ensure robustness.

* **Evaluation Metrics:**

Define evaluation metrics such as accuracy, precision, recall, F1-score, and mean Intersection over Union (mIoU) for comprehensive assessment of model performance.

**3.2.5 Deployment and Integration:**

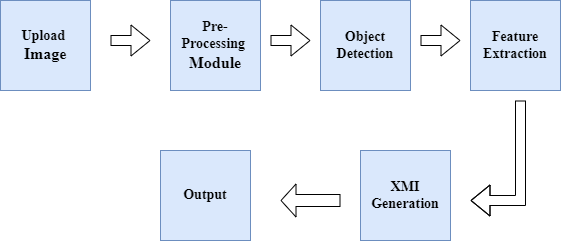
* **XMI Format Compatibility:**

Develop the software system to generate XMI files that adhere to the standard format accepted by existing UML tools.

* **Seamless Incorporation:**

Enable users to import the generated XMI files directly into their UML tool (StarUML).

## System Architecture



* **User Interface (UI) Layer:**

The UI Layer serves as the gateway for user interaction with the application. It encompasses visual components that facilitate the uploading of hand-drawn UML diagrams

* **Recognition Engine:**

Tasked with the analysis of hand-drawn UML diagrams, the Recognition Engine discerns UML entities and their interconnections. Its architecture is modular, comprising:

* **Preprocessing Module:**

This module standardizes incoming images, diminishes extraneous visual information, and rectifies any misalignment, setting the stage for accurate recognition.

* **Object Detection:**

Detects and localizes UML symbols using object detection algorithms.

* **Feature Extraction Module:**

Responsible for isolating characteristics from the refined images to pinpoint UML components such as classes, attributes, and associations, along with any textual notes.

* **Symbol Recognition Module:**

Employs advanced machine learning algorithms, like neural network classifiers, to identify UML components based on the isolated features.

* **OCR Integration:**

Incorporates Optical Character Recognition technology to detect and interpret text within the diagrams. The engine is designed for scalability and flexibility to accommodate diverse sketching styles and diagrammatic complexities.

* **Output Generation Module:**

This component translates the identified symbols and text annotations into a structured digital format, aligning with the requirements of prevalent UML tools. It produces XML-based output files that encapsulate both the graphical elements and their corresponding textual details, ensuring compatibility with standard UML modeling applications for effortless integration.

## 3.4 Use Cases

### Generate XMI File from Hand-Drawn UML Diagram

|  |  |
| --- | --- |
| Name | Generate XMI File from Hand-Drawn UML Diagram |
| Actors | User |
| Summary | The user uploads a hand-drawn UML diagram, and the software system processes it to generate an XMI file containing recognized UML symbols and textual annotations. |
| Pre-Conditions | The user has access to the software system.  Hand-drawn UML diagrams are available in digital format. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Post-Conditions | | The user obtains an XMI file representing the recognized UML symbols and textual annotations from the hand-drawn UML diagram. | | |
| Special Requirements | | None | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | The user opens the software interface. | | 2 | The software interface is displayed, prompting the user to upload an image. |
| 3 | The user uploads a hand-drawn UML diagram. | | 4 | The software processes the uploaded diagram using recognition algorithms. |
|  |  | | 5 | The software generates an XMI file containing recognized symbols and annotations. |
| 6 | The user retrieves the generated XMI file. | | 7 | The XMI file is provided to the user for download. |
| **Alternative Flow** | | | | |
| 3 | The user uploads an invalid or unreadable image. | | 4-A | The system responds with an error message: Invalid or unreadable image uploaded. |

**Conclusion:**

This chapter provides a comprehensive overview of the design and implementation plan for our project. It establishes the functional and non-functional requirements necessary for the system to operate efficiently and effectively. The chapter also highlights the importance of creating a custom dataset and the steps taken to annotate it accurately. The development and training of the recognition algorithms, along with the system architecture, are explained in detail to ensure clarity in understanding the project's approach. By automating the recognition and conversion of hand-drawn UML diagrams into digital formats, our system aims to streamline the software development process, making it easier for developers to update and maintain UML models.

# Implementation and Test Cases

**For each chapter provide a paragraph of introduction and in the end a paragraph of conclusions.**

## Implementation

Whatever implementation that you have done so far, please elaborate here.

Give clear details of the algorithms that were implemented along with the platform and the APIs which were used. **For FYP-1, this chapter can be changed to description of prototype developed.**

### Implementation of First Component/Algorithm

Write implementation of first component of your system here.

## **Test case Design and description**

**This section will be added in FYP-II.** Summarize the common attributes of test cases. This may include input constraints that must be true for every input in the set of associated test cases, any shared environmental needs, any shared special procedural requirements, and any shared case dependencies. The following scheme is recommended for describing test cases in detail.

### Sample Test case No.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **<Software component Name>** | | | | | |
| **<Reference>** | | | | | |
| Test Case ID: | | *Reference Number* | Test Date: | | *Date* |
| Test case Version: | | *Version number* | Use Case Reference(s): | | *Relation to use cases* |
| Revision History: | | *Refer to previous test case identity (if any)* | | | |
| Objective | | *Need and scope of the testing* | | | |
| Product/Ver/Module: | | *Refer to overall system being built and the place of this test case in it.* | | | |
| Environment: | | *Necessary and desired properties of the test environment. (hardware/software)* | | | |
| Assumptions: | | *Assumptions that might affect the testing process.* | | | |
| Pre-Requisite: | | *Necessary condition that needs to be fulfilled prior to the test case.* | | | |
| Step No. | Execution description | | | Procedure result | |
|  | *Events being tested.* | | | *Mention software response.* | |
| Comments: | | | | | |
| *Passed* *Failed* *Not Executed* | | | | | |

### Sample Test case No.2

.

.

.

## Test Metrics

Summarize here the common ground of attributes of test case metrics.

### Sample Test case Matric.No.1

|  |  |
| --- | --- |
| Metric: | Purpose |
| Number of Test Cases: | Total number of test cases that you have developed for your system. |
| Number of Test Cases Passed: | The number of test cases that successfully passed |
| Number of Test Cases Failed: | The number of test cases that failed |
| Test Case Defect Density: | (No of test cases failed \* 100)  No of test cases executed |
| Test Case Effectiveness: | No of defects detected using test cases \*100  Total number of defects detected |
| Traceability Matrix: | Traceability is the ability to determine that each feature has a source in requirements and each requirement has a corresponding implemented feature. |

### Sample Test case Metric.No.2

### Sample Test case Metric.No.3

.

.

.

# Experimental Results and Analysis

**This chapter will be added in FYP-II.** Give proper analysis and discussion of experimental results (in plain English text) along with tables of results. **For each chapter provide a paragraph of introduction and in the end a paragraph of conclusions.**

# Conclusion and Future Directions

**This chapter is mandatory.** Give conclusions and summary of the work done. What were your findings and what were the results? Discuss in detail whether the scope of your project was entirely covered or not and whether the objectives of the project were met or not. What challenges did you face and what has been left out and why?

Sum up all the conclusions of all the chapters here to make a conclusion chapter. Do not repeat any text, just summarize it in different words.

Give recommendations for future work also. How your project can be further enhanced or improved? Future recommendations if someone wants to work on it. **For FYP-1 it is mandatory to list down a plan of the work to be done for FYP-2.**

# References

# [1] B. Karasneh, and M. Chaudron, “Extracting UML Models from Images,” in Proc.

# International Conference on Computer Science and Information Technology (CSIT), pp. 1-15,

# 2013. DOI: 10.1109/CSIT.2013.6588776.

# [2] B. Schafer, “Recognizing Hand-drawn Diagrams in Images,” Ph.D. dissertation, Univ. of

# Mannheim, Mannheim, 2023.

# [3] M. Axt, “Transformation of sketchy UML Class Diagrams into formal PlantUML models,”

# Final Project, Main field of study: Computer Engineering BA ©, Credits: 15, Semester/year:

# Spring 2023, Supervisor: R. Jolak, Examiner: F. Dobslaw, Course code/registration number:

# DT133G, Programme: Software Engineering

# Appendix

## Appendix A: Guidelines

This section should include all supporting information from the project that was not included in the body of the report.  You should include surveys, complex statistical calculations, certain detailed tables and other such information in an appendix.  The information presented in this section is important to support the work presented in the body of the report but would make it more difficult to read and understand if presented within the body of the report.

Cite the appendix items in the report narrative (write "see Appendix A") and organize appendices (e.g., Appendix A, Appendix B,

Any tables, figures, forms, or other materials that are not totally central to the analysis but that need to be included are placed in the Appendix.

## Appendix B: Heading of Sample Appendix B

Following is a sample code with “code” style format.

Void SampleFunction(){

Print “Hello World.”;

}

# Formatting Guidelines

This document also serves as style guide for final year project reports. In order to give a similar high-quality appearance to all final year software project reports this template uses a collection of predefined Microsoft Word formatting styles. **These styles should be used without modification or replacement.** Font in the document is ***“Time New Roman”.*** This template provides following styles:

* **Title** – the main title style
* **Title2** – the subtitle style
* **Body Text** – style for paragraphs
* **Caption** – the style for a figure or table caption
* **Table Description** – the style for description of table, it must be added after caption.
* **Figure Description** - the style for description of figure, it must be added after caption.
* **Code** – the style for program source code

**int x** = 10; // Writing important code

* **Table Header Row** – Style for the header row of table
* **Table Grid** – the style for the data rows in the tables
* **Reference** – The style for references
* **Bullets** – The style for the bullet lists
* **Numbered** **List**– Style for numbered lists

All Heading styles with different level numbers are listed below.

# Heading 1

## Heading 2

### Heading 3

#### Heading 4

##### Heading 5

###### Heading 6

Heading 7

Heading 8

Heading 9

## Tables and Figures

Tables and figures should be centered horizontally. The caption button should be used to insert caption for both the figures and tables. All figures and tables must be numbered properly. Always refer to tables and figures according to their numbers. A table or figure can be cited as follows: ‘see Table1’ or ‘as shown in Table1’. The caption of table should be centered above the table and figure caption should be centered below the figure. Place the tables/figures close to their reference. Use “Table Header Row” and ‘Table Grid’ style for table’s header and data rows respectively. It is compulsory to provide brief description of table/figure after its caption. Styles for table and figure descriptions are “Table Description” and “Figure Description” respectively.

Press Ctrl+Shift+S to see list of styles mentioned above. Figure 1 shows the Apply Style window displaying the list of styles. Select any text then press Ctrl+Shift+S, the Apply Style window will show you the current style applied on that text and if required, you can change the style by selecting any other style from the “Style Name” dropdown.

This is brief description of above figure.

Figure 1: List of Styles

Table 1: This is Sample table caption

This is brief description of following Table.

|  |  |  |  |
| --- | --- | --- | --- |
| Header row | Header row | Header row | Header row |
| Row1 col1 | Row1 col2 | Row1 col3 | Row1 col4 |
| Row2 col1 | Row2 col2 | Row2 col3 | Row2 col4 |

Table 2: This is Sample table caption

This is brief description of following Table.

|  |  |  |  |
| --- | --- | --- | --- |
| Header row | Header row | Header row | Header row |
| Row1 col1 | Row1 col2 | Row1 col3 | Row1 col4 |
| Row2 col1 | Row2 col2 | Row2 col3 | Row2 col4 |

## Equations

Use equation editor to write equations in this report. Use last button of the custom tool bar to invoke equation editor. Similar to tables and figures, equations should also be aligned centered horizontally. Number all equations and insert them in parenthesis. Below is a sample equation and its reference number. An equation can be referenced like this: ‘it is clear from (1)’.

 (1)

## Header/Footer

Notice the headers in this document, before Introduction (i.e. the main content of this document) page numbers are in roman numerals. The page numbers of the actual content start with Arabic numerals i.e. 1, 2, 3 and so on. All of the **odd numbered pages** contain title of your project while the **even numbered pages** contain the section heading (i.e. chapter’s name) in the headers.

## Other Formatting Guidelines

* Keep 2-4 GUIs in one page. Consume as much space as possible. Do not leave most of page blank unnecessarily.
* Do not break tables (or use cases) in multiple pages unless the table is too large to fit in one page.
* Re-arrange the content i.e., text, images, and tables properly to meet above two guidelines.

## References

Always refer to the source of information by inserting the reference number in square brackets like this [5]. The reference numbers can either be added at the end of the sentence or within the sentence without changing the punctuation of sentence. A reference can also be cited as follows: ‘as Ruskey [2] mentioned’. List each source only once on your reference page.



Figure 2: IEEE Reference style

This figure represents the styling information for adding references in IEEE format

**Following is a list of sample reference for various typed of sources in IEEE format.**

1. P.M. Morse and H. Feshback, *Methods* of *Theoretical Physics*. New York: McGraw Hill, 1953. **//Format for Book**
2. S.K. Kenue and J.F. Greenleaf, “Limited angle multifrequency diffiaction tomography,” *IEEE Trans. Sonics Ultrason*., vol. SU-29, no. 6, pp. 213-2 17, July 1982. **//Format for Journal Article**
3. B. Tsikos, “Segmentation of 3-D scenes using multi-modal interaction between machine vision and programmable mechanical scene manipulation,” Ph.D. dissertation, Univ. of Pennsylvania, BCE Dept., Philadelphia, 1987. [Add if applicable: University Microfilms, Inc., University of Michigan, Ann Arbor, Michigan.] **//Format for Dissertation or thesis**
4. R. Finkel, R. Taylor, R. Bolles, R. Paul, and J. Feldman, “An overview of AL, programming system for automation,” in *Proc. Fourth Int. Joint Conf Artif. Intell*., pp. 758-765, Sept. 3-7, 1975. **//Format for Proceedings paper**
5. “Technology threatens to shatter the world of college textbooks, *The Wall Street Journal*, vol 91, pp. Al, A8, June 1, 1993. **//Format for Newspaper article**
6. R. Cox and J. S. Turner, “Project Zeus: design of a broadband network and its application on a university campus,” Washington Univ., Dept. of Comp. Sci., Technical Report WUCS-91-45, July 30, 1991. **//Format for Technical Report**
7. M. Janzen, *Instant Access Accounting*. Computer software. Nexus Software, Inc IBM-PC, 1993. **//Format for** **Software**
8. Fuminao Okumura and Hajime Takagi, “Maglev Guideway On the Yamanashi Test Line,” *http://www.rtri.or.jp/rd/maglev2/okumura.html*, October 24, 1998. **//Format for** **World Wide Web** (give author and title if named)
9. “AT&T Supplies First CDMA Cellular System in Indonesia,” http://www.att.com/press/1095/951011.nsa.html, Feb 5, 1996. **//Format for World Wide Web**