**National University of Computer & Emerging Sciences**

**Karachi Campus**



**Automated Rubik’s Cube Solver Using Computer Vision**

**Project Proposal**

**Artificial Intelligence**

**Section: E**

**Group Members:**

**22k-4350 Muhammad Hasheem**

**22k-4163 Imaddudin Muhammad**

**22k-4315 Ali Soban**

## 1. Introduction

This project aims to develop an automated Rubik’s Cube solver that combines computer vision with an efficient cube-solving algorithm. The system will use a webcam to capture the six faces of a physical Rubik’s Cube, process the images with OpenCV to determine the color configuration, and then compute the optimal sequence of moves required to solve the cube. The solution moves will be derived from a Python-based solver that implements face rotation functions and search routines.

## 2. Existing System

Many Rubik’s Cube simulators and solvers are available online. However, most require manual input of the cube configuration. Open-source projects on GitHub often offer cube solvers in various languages, yet few integrate real-time computer vision for state detection. This project addresses the gap by seamlessly combining automated cube scanning with an intelligent solver, reducing user input and increasing overall usability.

## 3. Problem Statement

Current Rubik’s Cube solvers either rely on manual configuration or are limited to digital simulations. This approach is error-prone and does not reflect real-world scenarios. Our project seeks to overcome these limitations by:

* Automatically detecting the cube’s current configuration using OpenCV.
* Integrating a robust Python-based solver that uses advanced cube rotation functions.
* Providing an interactive solution that guides the user through the solution moves.

Overall, our project will fulfil the real world requirements making it efficient and creative.

## 4. Proposed Solution

Our solution will consist of two main components:

1. Cube State Detection via Computer Vision:
   * Image Capture & Preprocessing: Use OpenCV to capture live images of the Rubik’s Cube from a webcam.
   * Color Recognition: Apply color-detection techniques to identify each sticker’s color accurately, even under varying lighting conditions.
   * State Reconstruction: Map the detected colors to the corresponding cube faces to reconstruct the cube’s current state.
2. Cube Solving Algorithm:  
   * Cube Representation: Implement a Rubik’s Cube data structure in Python.
   * Rotation Functions & Search: Utilize custom rotation routines to simulate cube moves.
   * Solution Generation: Integrate a search strategy using efficient heuristics to compute the sequence of moves needed to solve the cube.

## 5. Salient Features

* Automated Detection
* Efficient Solver
* Real-Time Feedback:  
   Integration of live video feedback so users can see the cube state.
* User-Friendly Interface:  
   A command-line and/or GUI interface that guides the user through scanning, solving, and move execution.
* Extensibility

## 6. Tools & Technologies

* Programming Language:  
   Python
* Computer Vision:  
   OpenCV for image capture  
    
   Libraries:  
  + NumPy for array operations.
  + SciPy
  + Kociemba’s algorithm
* Development Environment:  
   Any modern IDE on Windows or Linux with access to a webcam.