Vision-aided Mechanical Design for an Autonomous Rubik's Cube Solver

Team Members

Abhinav Chalise (THA077BEI003)
Nimesh Gopal Pradhan (THA077BEI026)
Nishan Khanal (THA077BEI027)
Prashant Raj Bista (THA077BEI032)

Supervisor

Er. Dinesh Baniya Kshatri
Assistant Professor

Department of Electronics and Computer Engineering
Institute of Engineering, Thapathali Campus

March 2024

Presentation Outline

- Introduction
- Motivation
- Problem Statement and Objectives
- Scope of Project
- Methodology
- Results
- Applications
- Future Enhancements
- Conclusion
- Timeline
- Project Budget
- References

Motivation

- Rubik's Cube Combination
  - About Forty-three quintillion States (43,252,003,274,489,856,000)
- God's Number
  - Upper Bound to Solve any State = 20

3x3x3 Rubik's Cube

Introduction

- A mechanical system to solve scrambled 3x3x3 Rubik's Cube.
- A GUI system to virtualize the state of the Rubik's Cube.
- Leverages YOLOv8 for detection of colors in the cube.
- Uses the Kociemba's Algorithm for getting cube solutions.

Problem Statement and Objectives

- Problem Statement:
  - Enhancing Rubik's Cube-solving from tedious to enjoyable experience.

- Objectives:
  - To design and build a mechanical system capable of solving scrambled 3x3x3 Rubik’s Cube.

  - To build a system capable of virtualizing a physical Rubik’s Cube by capturing initial scrambled state of the cube.

Scope of Project

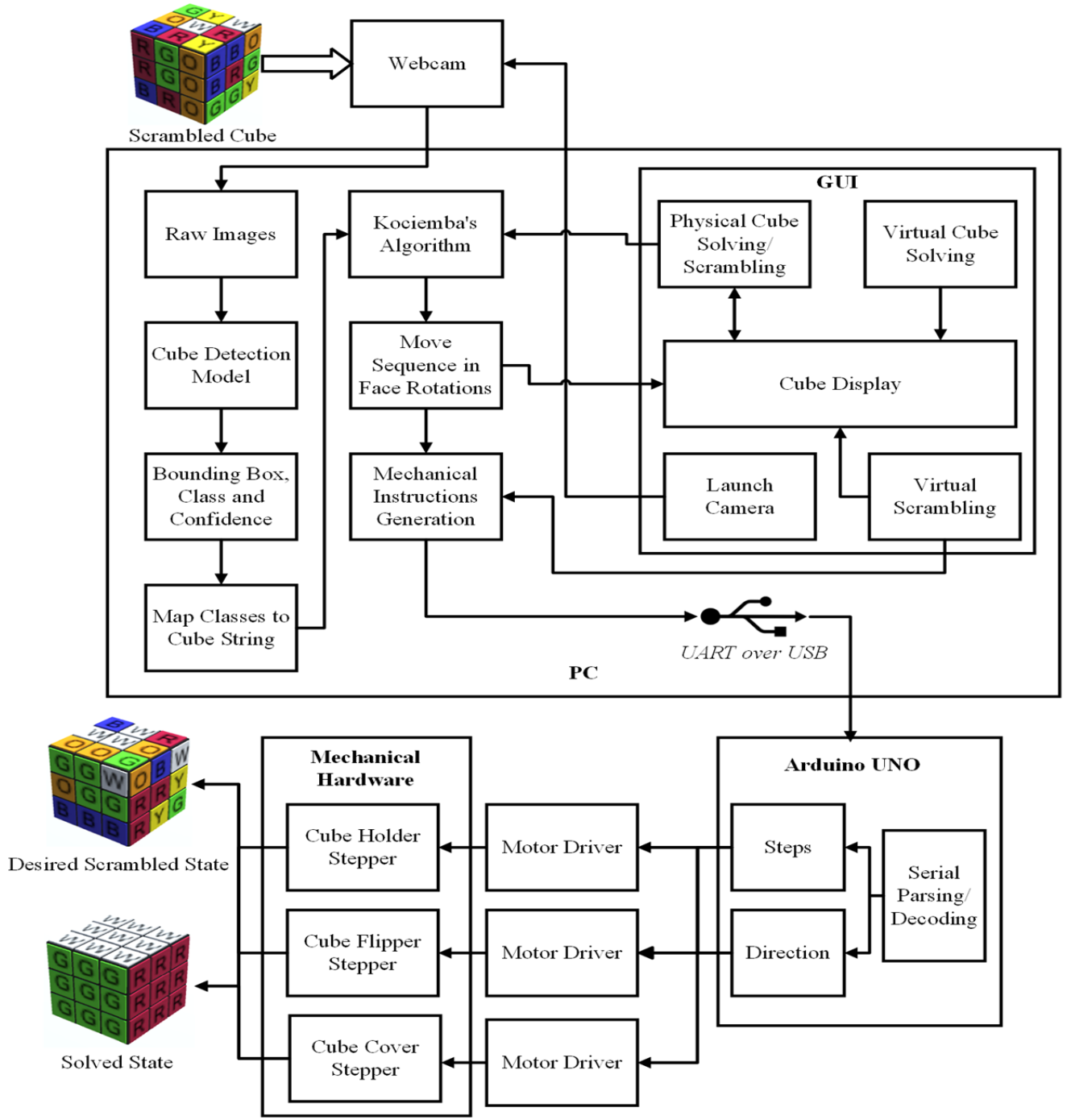
- Project Capabilities:
  - Vision Guided Cube Solving
  - Mechanical Design for precise cube manipulation
  - GUI for virtualized representation of real-world cube

- Project Limitations:
  - Speed limitation due to single degree of freedom in cube-face rotations
  - Only capable of solving 3x3x3 Rubik’s cube
  - Exclusive use of Kociemba's Algorithm
  - Requires manual calibration being open loop system

Applications

- Educational Tool
  - Mechatronics Demonstration, Algorithmic Learning
- Entertainment and Display
  - Technological Exhibits, Pop Culture Integration
- Commercialization Potential
  - Consumer Product, Educational Kit
- Quality Control and Testing
  - Coordinating Computer Vision and Mechanical Movements

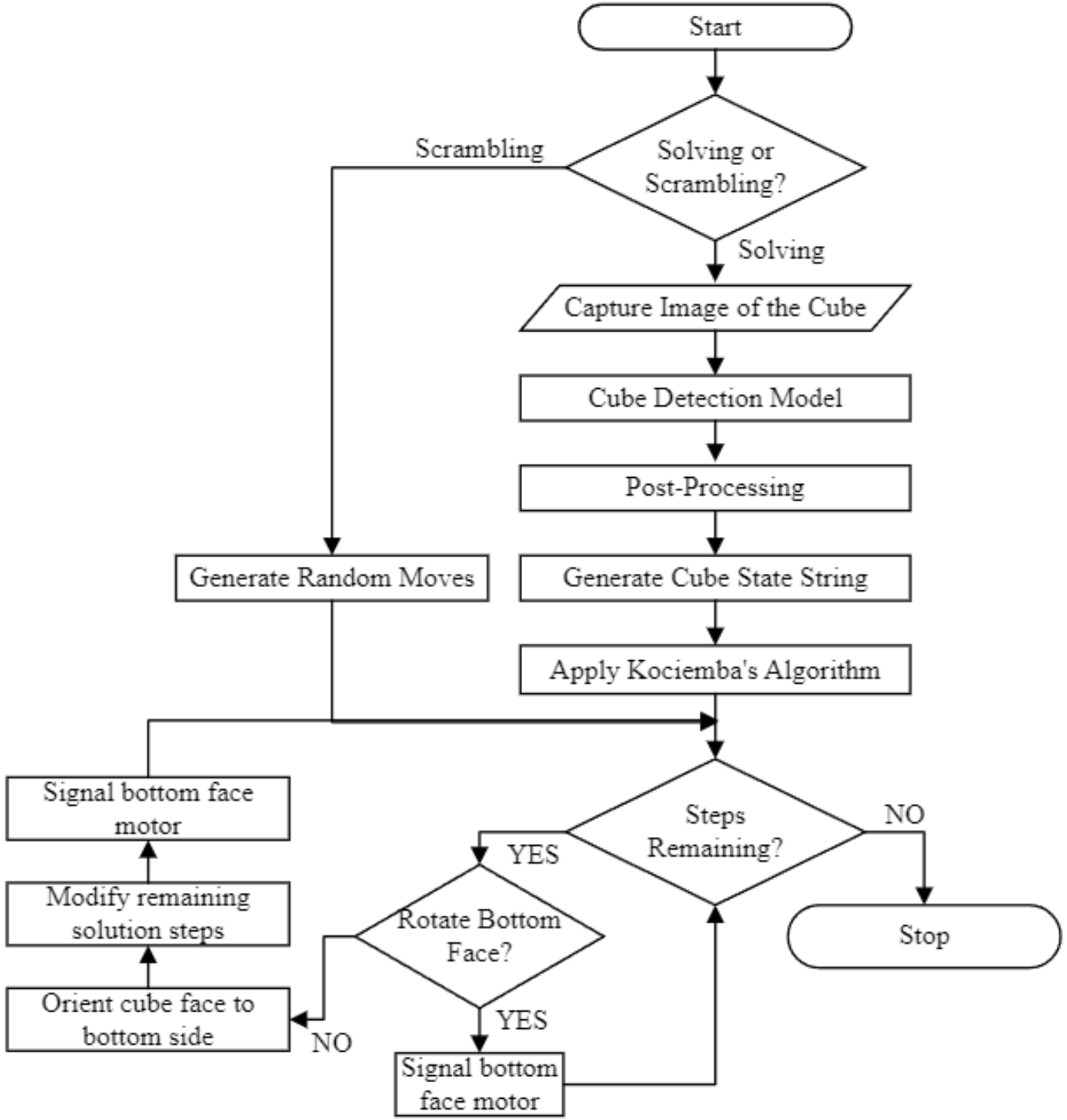
Methodology - [1] (System Block Diagram)



Methodology - [2] (Working Principle)

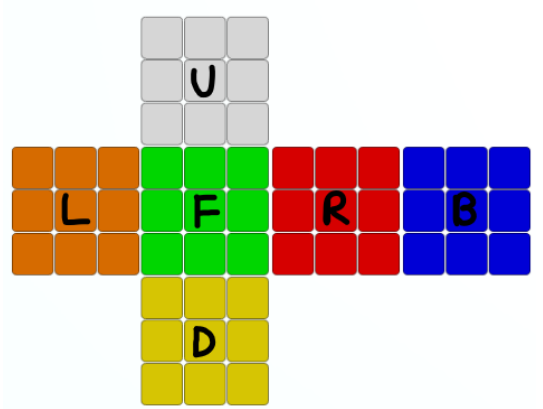
- Cube image captured through webcam.
- Object Detection using YOLOv8 model
- Generate string representation of cube
- Cube virtualized in GUI
- Generate mechanical instructions after Kociemba's algorithm to solve
- Or generate mechanical instruction randomly for scrambling
- Or interact with physical cube from virtual cube in GUI

Methodology - [3] (Flow of the System)

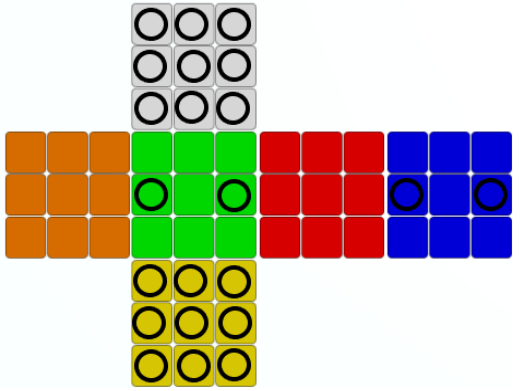


Methodology - [4] (Kociemba's Method)

- Kociemba's algorithm splits the solution into two distinct phases.
- Phase One
  - Transform any scrambled cube into a specific intermediate state in subset H.
  - Subset H contains 20 billion positions,
  - Corner and edge pieces are correctly oriented, edge pieces of middle layer remains in middle layer
- Phase Two
  - Use a limited set of moves A = {U, U2, U′ , D, D2, D′, R2, L2, F2, B2} to fully solve the cube from the intermediate state.

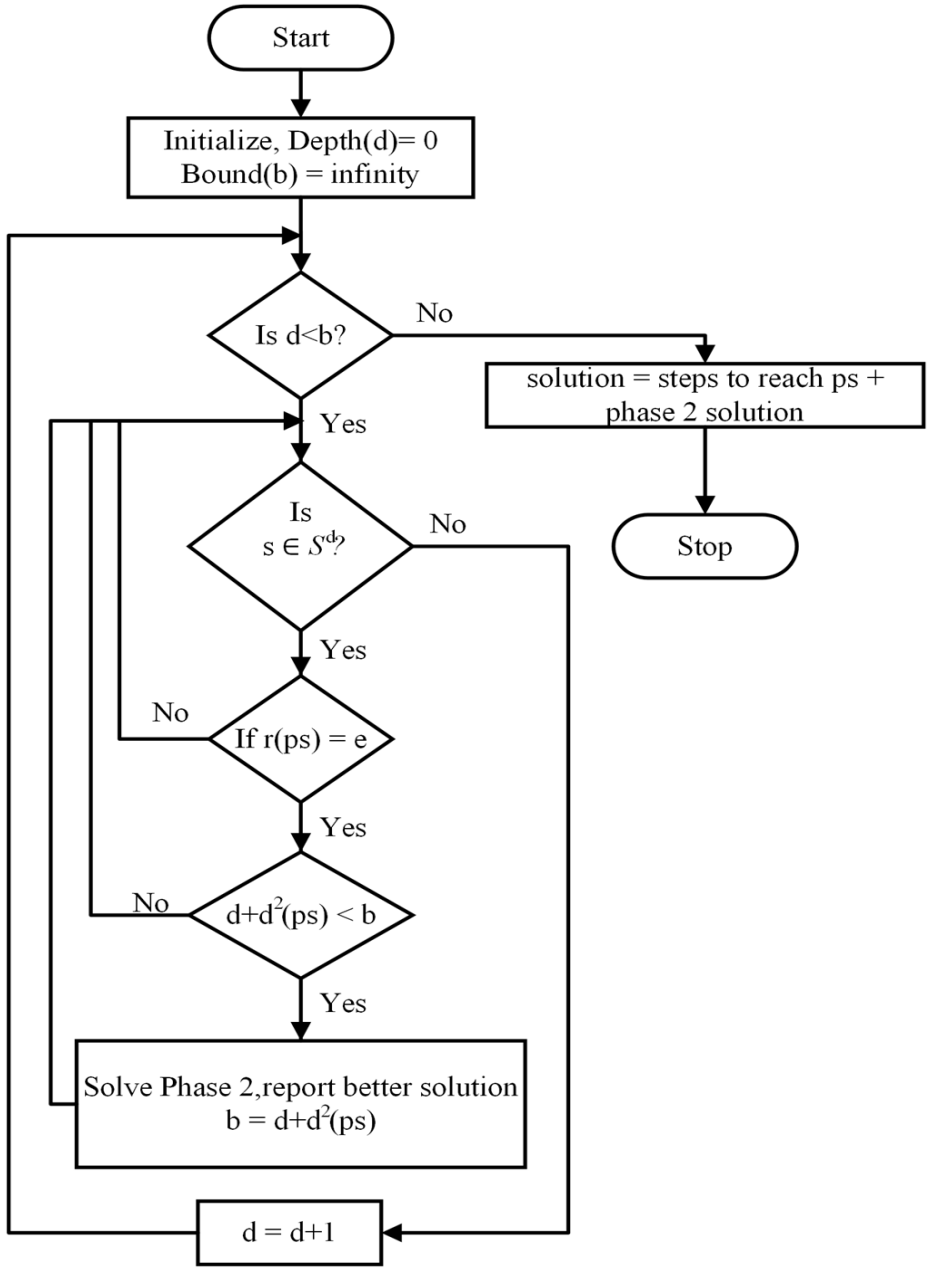


2D cube showing face label



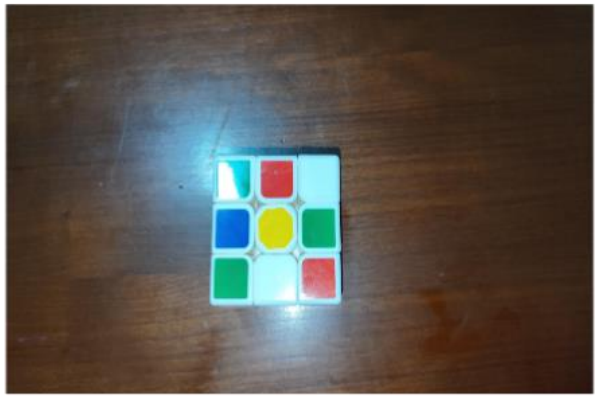
2D cube showing reference orientation

Methodology - [4] (Kociemba's Method)

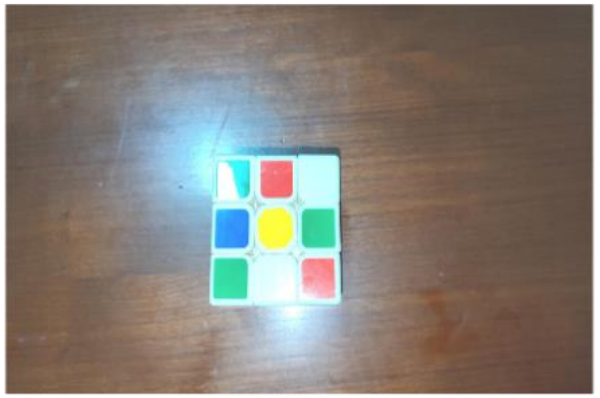


Symbol	Meaning
d	depth
b (= 24 practically)	bound
S^d	Set of all states at depth d
ps	Present State

Methodology - [6] (Dataset Augmentation)

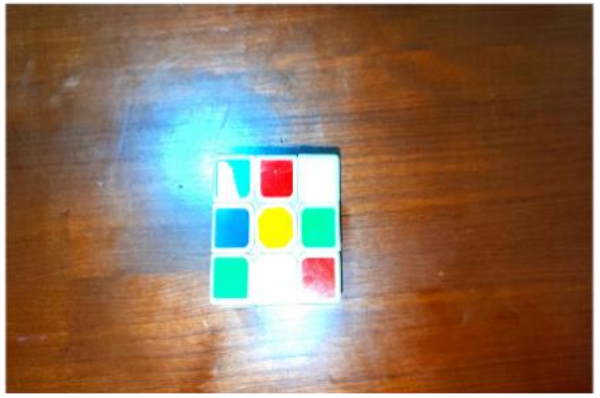


Raw Image



Increased Saturation

Decreased Contrast



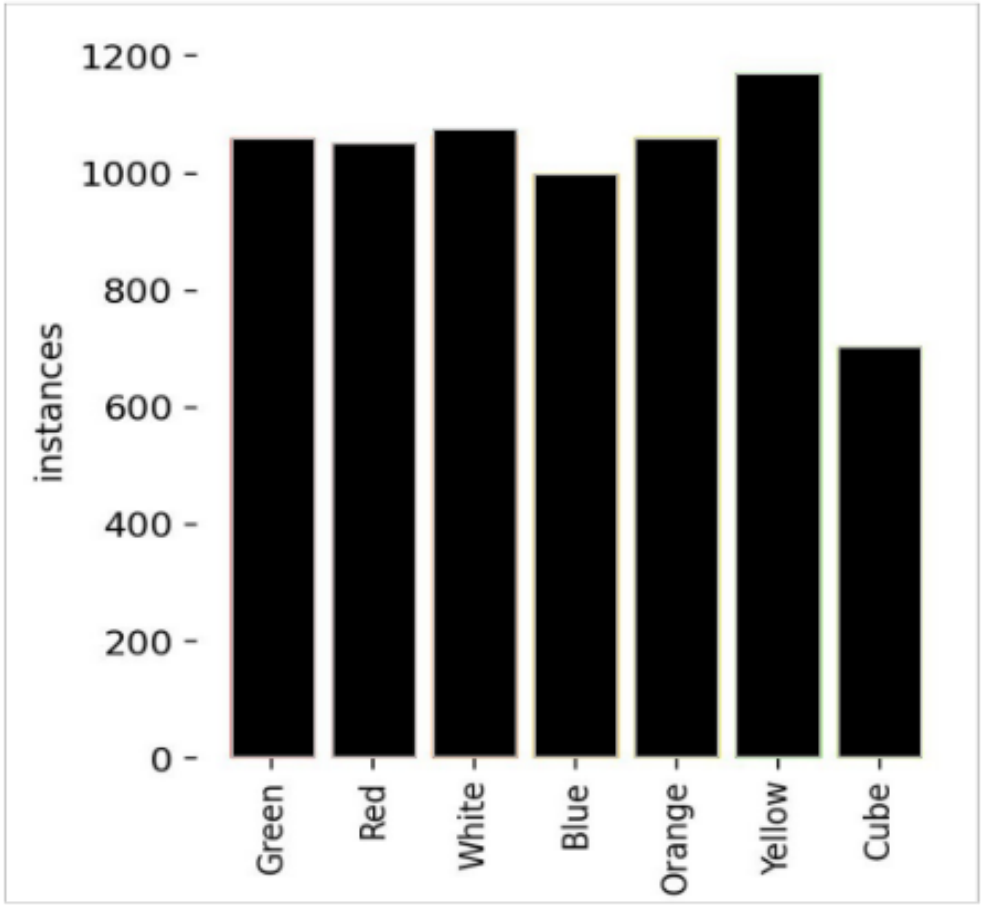
Increased Saturation, Contrast, Brightness

Methodology - [7]
(Pre-training and Fine-tuning Dataset)

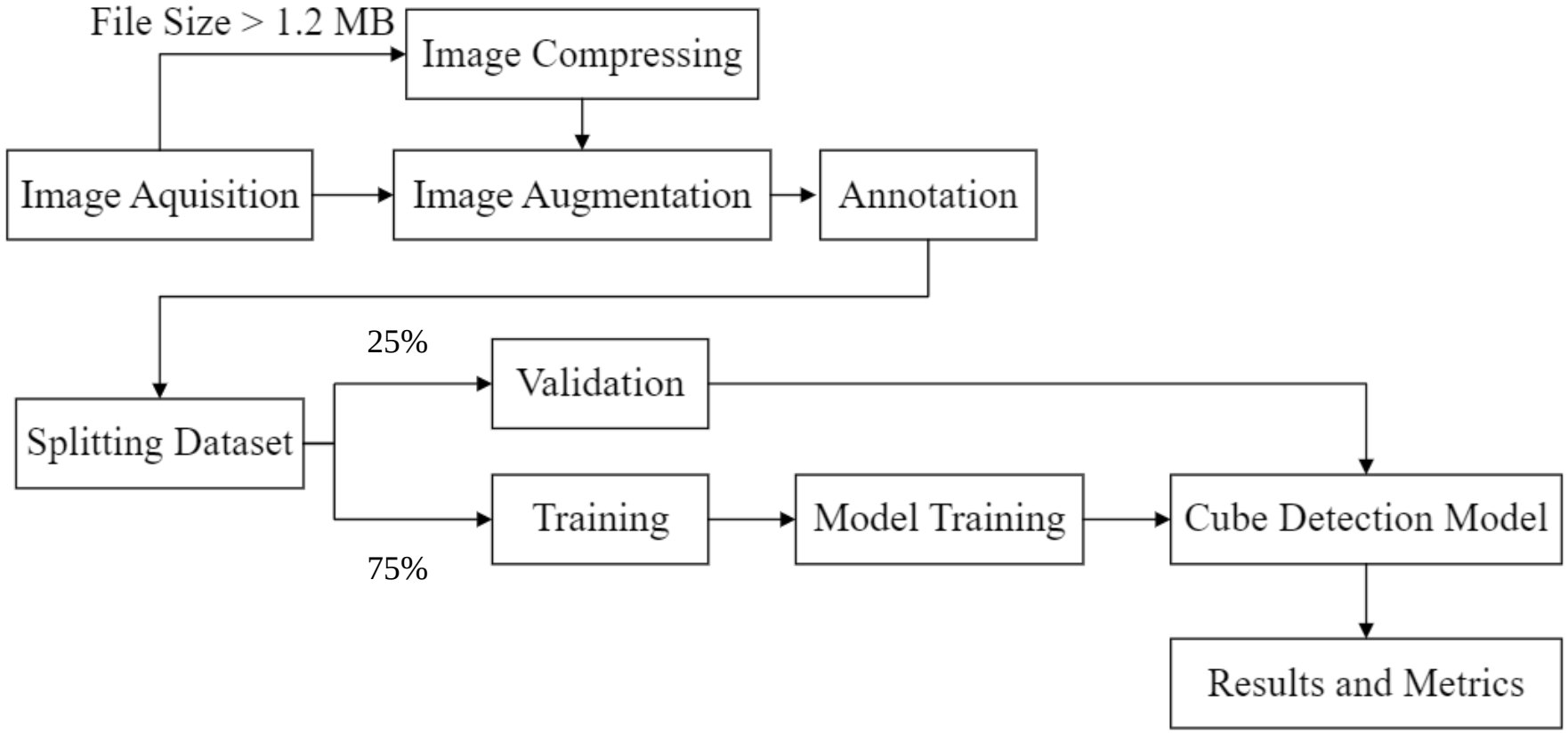
Pre-training Dataset

- Model was pre-trained on Common Objects in Context (COCO dataset)
- Total Images: 330,000
- Total Classes: 80
- Training Images: 118,287
- Validation Images: 5,000

Fine-tuning Dataset



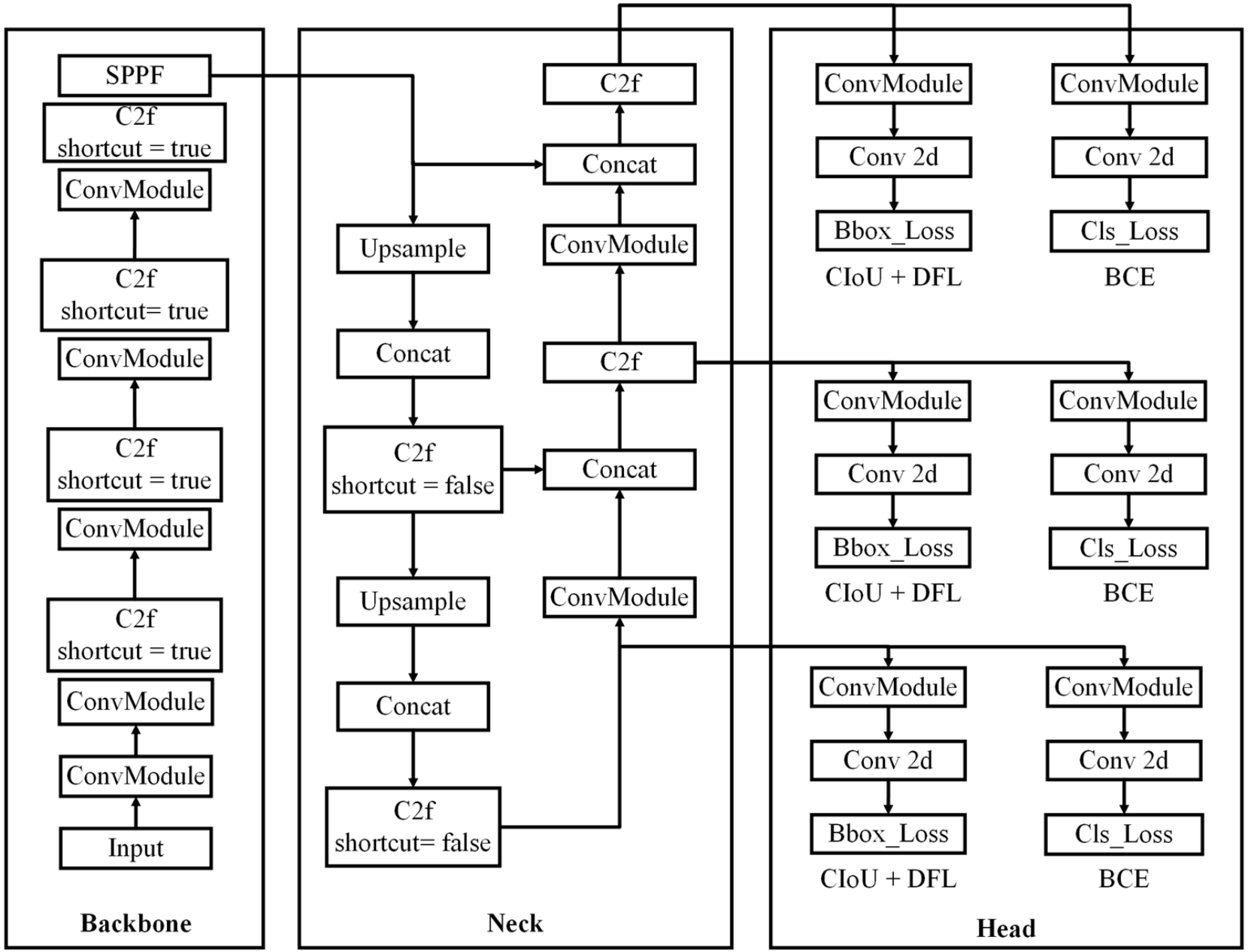
Methodology - [5] (Training Pipeline)



Methodology - [8] (Hyperparameter Values)

Hyperparameters	Values
"epochs"	70
Initial Learning rate ("lr0")	0.000909
Image size ("imgsz")	1120 (px)
"model"	"yolov8n.pt" (pre-trained)
"optimizer"	"auto" (AdamW)
"momentum"	0.9

Methodology - [9] (YOLO's Architecture - [1])



Methodology - [9]
(YOLO’s Architecture - [2])

- Backbone
  - Shrinks image size, making feature processing easier and more efficient.
  - C2F Layer incorporates "shortcut" that skips connection to preserve spatial detail and captures abstract features
  - SPPF Block uses MaxPool2d of various scales to capture features of objects of all sizes.
- Neck
  - The neck refines backbone features in YOLO architecture
  - Up sampling and concatenation increases spatial resolution and feature scale

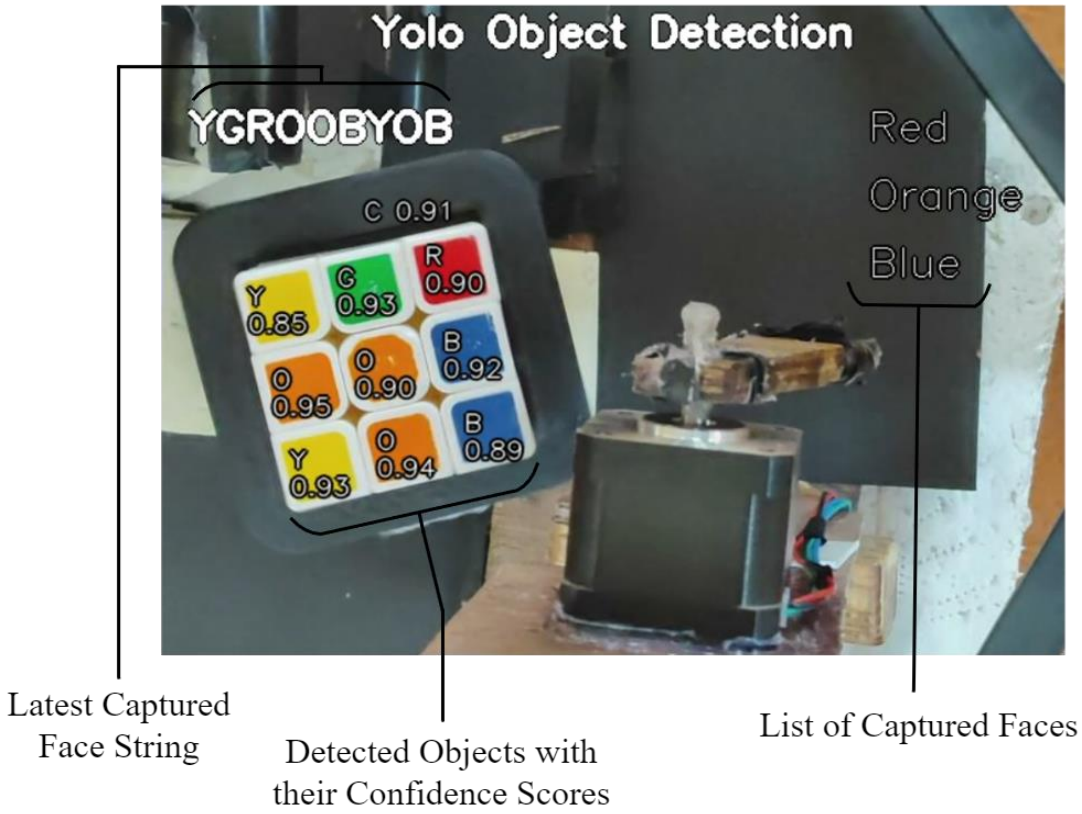
Methodology - [9]
(YOLO's Architecture - [3])

- Head
  - Uses processed features from the backbone and neck for detection.
  - Box Loss and Class Loss ensure accurate size and class prediction.
  - CIoU+DFL for box alignment; BCE for class accuracy.

- Non-Maximum Suppression
  - Enhances clarity by removing redundant detections, focusing on confidence and IoU criteria.
  - Ranks boxes by confidence; suppresses those overlapping significantly, especially over IoU threshold (e.g., 0.5).

Methodology - [10] (Cube Detection From YOLO)

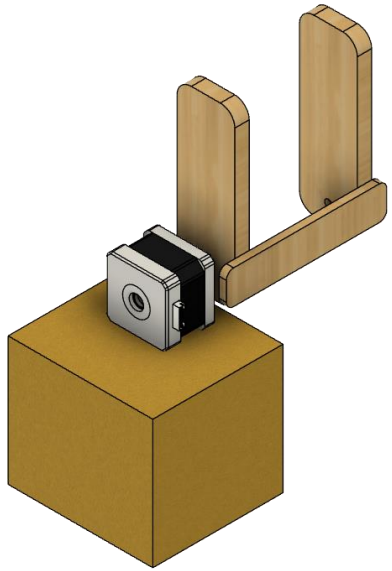
- Initial state capture with camera
  - Camera is positioned at the top to capture faces without obstruction
- State analysis with YOLO
  - YOLOv8 model is used to process the captured image and extract the color of cube faces
  - Detections and their confidence score is displayed on screen



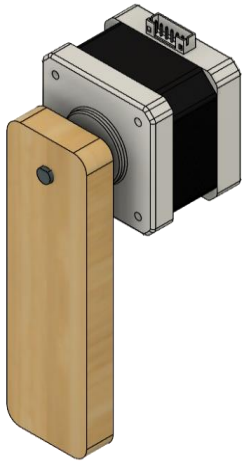
OpenCV Application with YOLOv8n object detection model

Methodology - [11]
(CAD Design of Hardware)

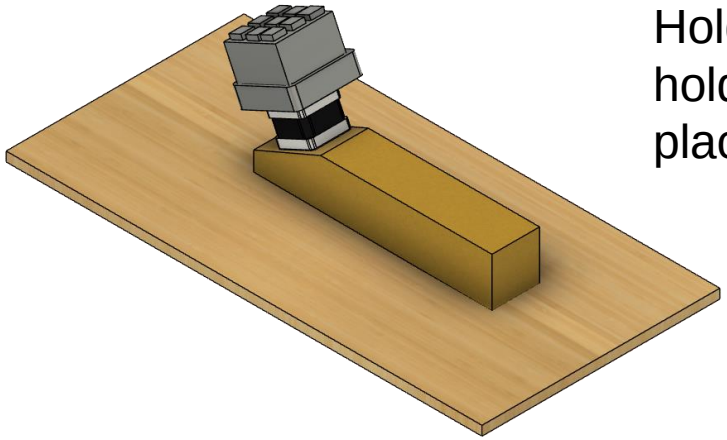
Cube Holder
Holds Cube,
Rotates bottom layer



Cube Cover
Covers Cube for
Bottom Face Rotation



Cube Flipper
Flips the cube



Cube Holder Base
Holds the cube holder and motor in place.

Methodology - [12] (Hardware Implementation)

- Cube holder holds the Rubik's Cube between two beams
- Stepper Motor rotates the flipper 360°; flipping the cube
- Cover holds top two layers and rotating only the bottom layer
- Cube Holder Base is inclined at 10° for consistent flipping

Methodology - [13] (Calibration of System)

- Calibrating Cover
  - Position Cover parallel to the motor base
  - Rotate Cover 73° away from the cube
- Calibrating Cube Holder
  - Insert a metal piece into the styrofoam base
  - Rotate cube holder clockwise until it makes contact with metal piece
  - Remove metal piece
  - Rotate cube holder 30° anticlockwise direction

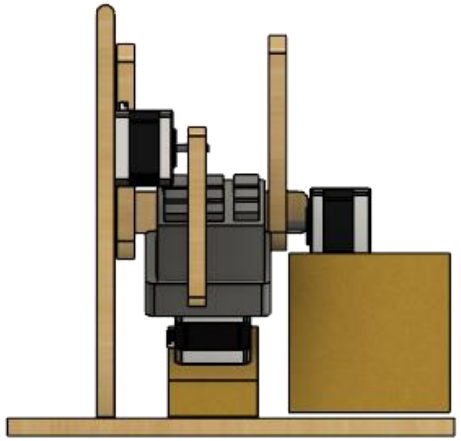
Methodology - [14] (Hardware Used)

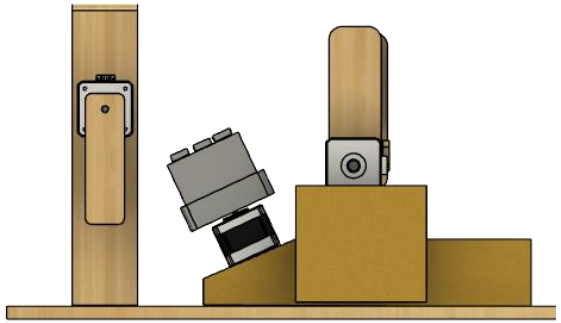
Hardware	Purpose
Stepper Motor (STH-39H112-06) (NEMA17-42HS40)	Rotate bottom layer of cube Flipping the cube
Motor Driver (A4988)	Act as switch to help motor rotate Contains special circuitry to control speed of motor
Microcontroller (Arduino UNO)	Controlling motor and running software
Camera (Mobile's Camera)	Capture image of Rubik's cube

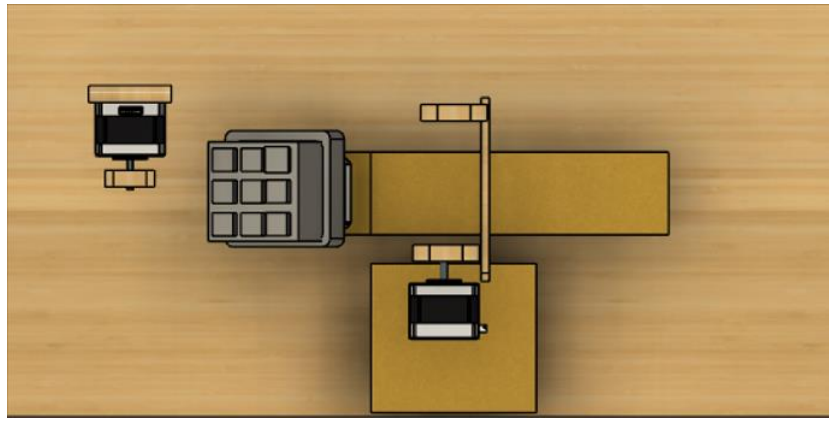
Methodology - [15] (Software Used)

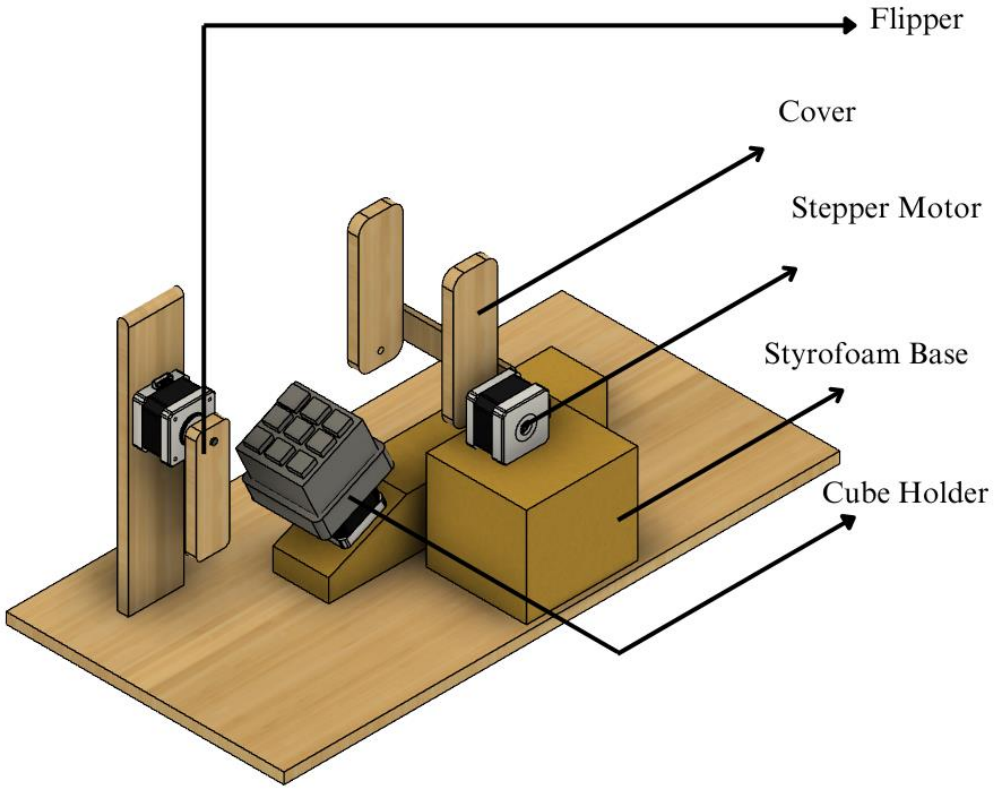
Software	Purpose
You Only Look Once(YOLOv8n)	Real Time Cube Face Detection
Autodesk Fusion360	3D modelling and mechanical designing
Unity Hub (Game Engine)	GUI development
Kociemba's Algorithm	Algorithm for solving the cube

Result - [1]
(Mechanical Design (3D-Model))

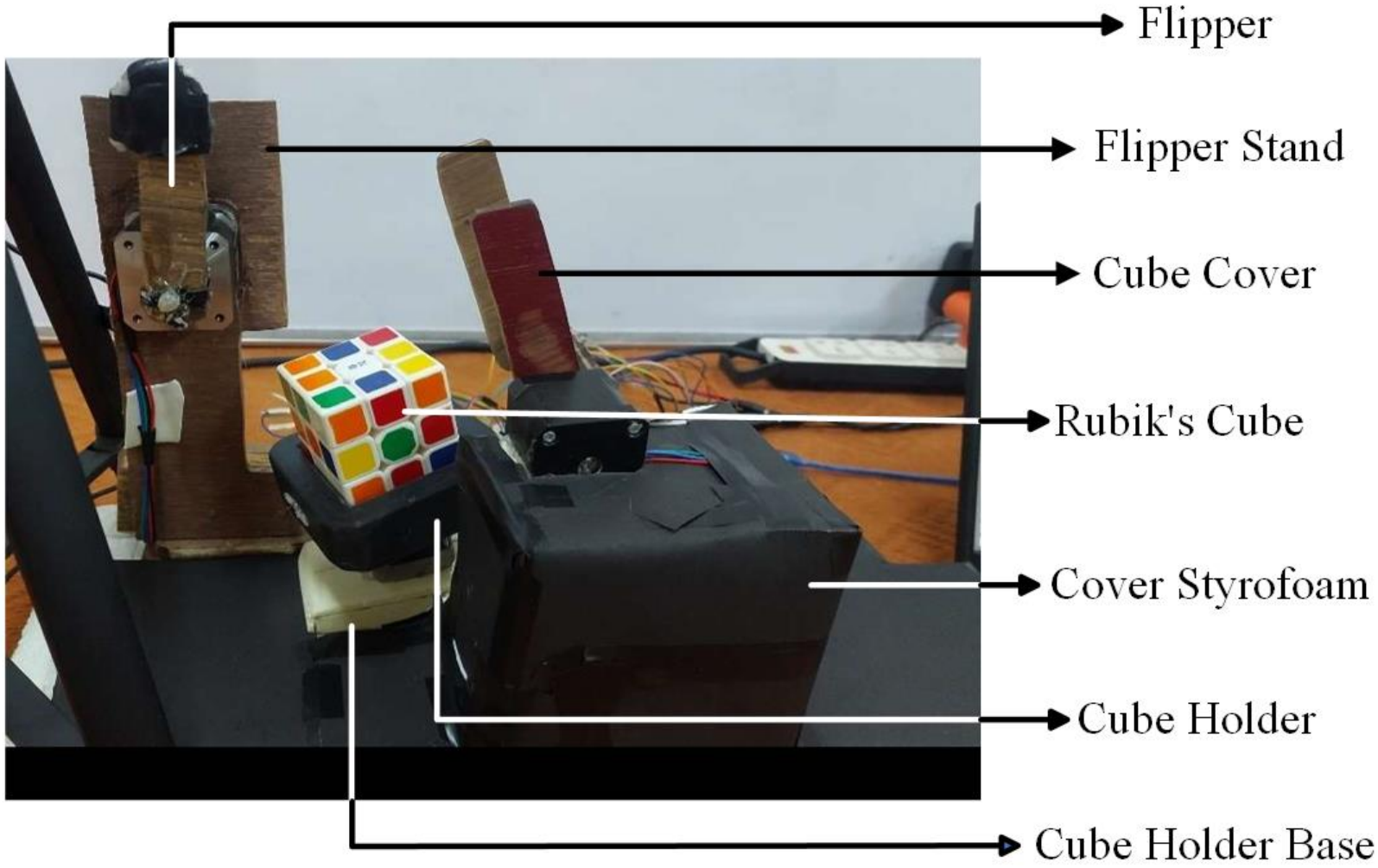








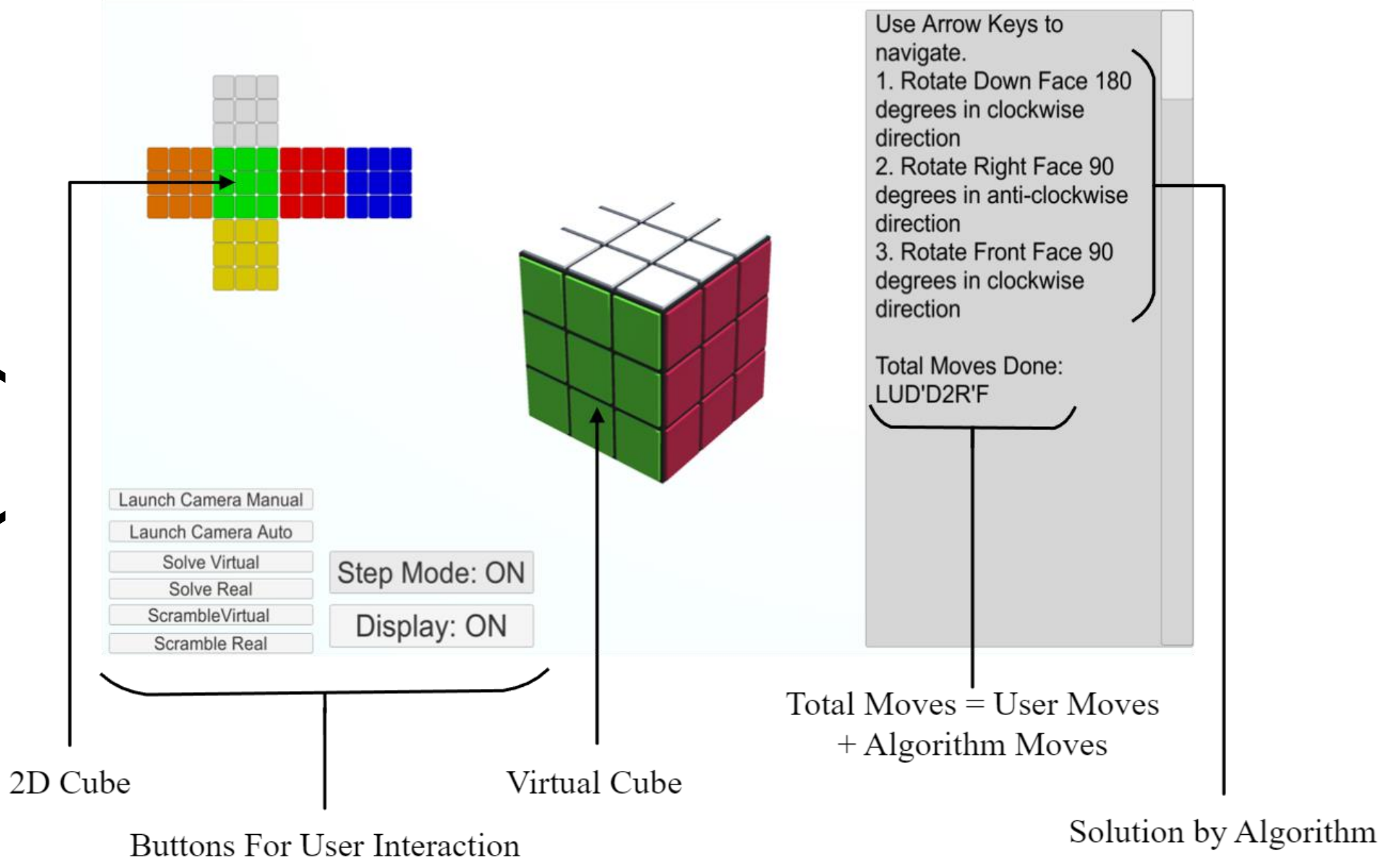
Result - [2]
(Actual Assembled Model)



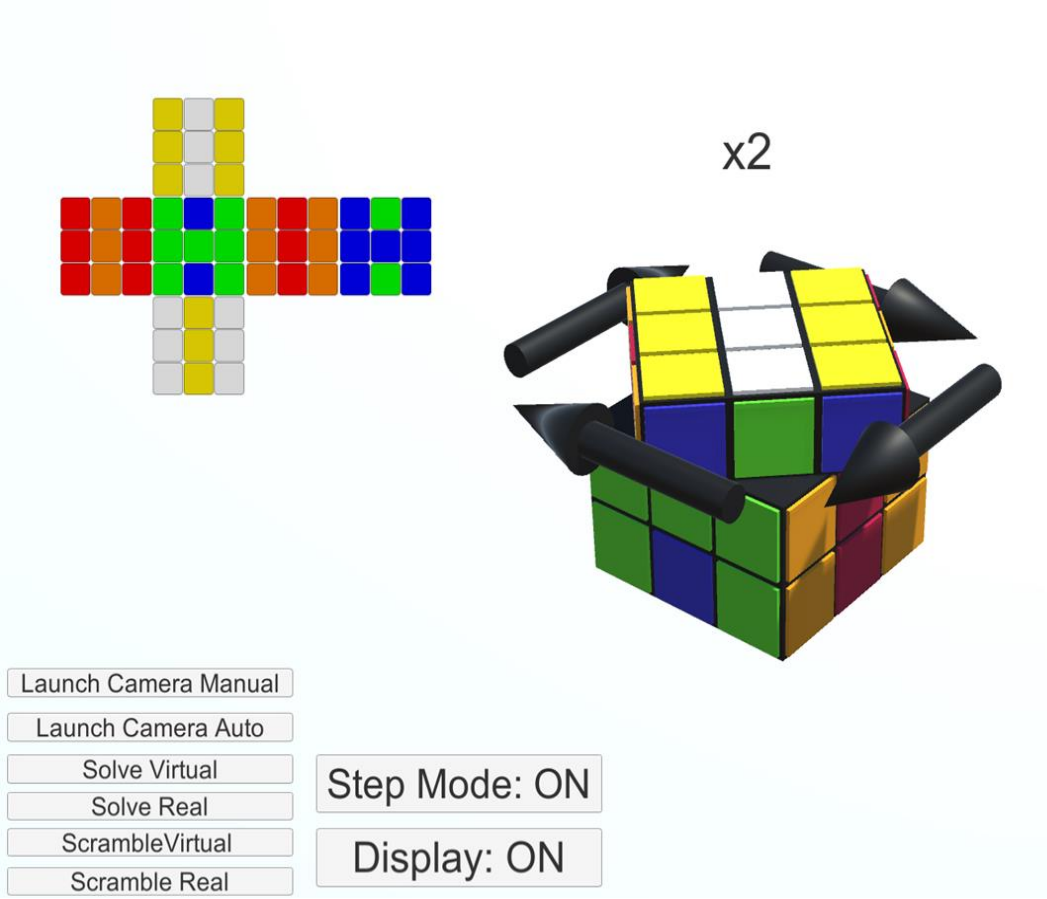
Result- [3]
(Comparison of 3D Design and Actual Model)

Dimensions	Actual Size	3D Model Size
Total Length of Model	47 cm	50 cm
Total Breadth of Model	25 cm	25 cm
Total Height of Model	23 cm	23 cm
Cover Length	11.5 cm	12 cm
Cover Styrofoam Height	12 cm	8.904 cm
Cover Styrofoam Width	11.5 cm	9 cm
Flipper Length	9.5 cm	10 cm
Flipper Stand Height	24.8 cm	23 cm
Cube Base Height	4 cm	5.109 cm
Angle of Inclination of Cube Base	10 degrees	33 degrees
Cube Base Length	19.5 cm	25.307 cm

Result - [4] (GUI)

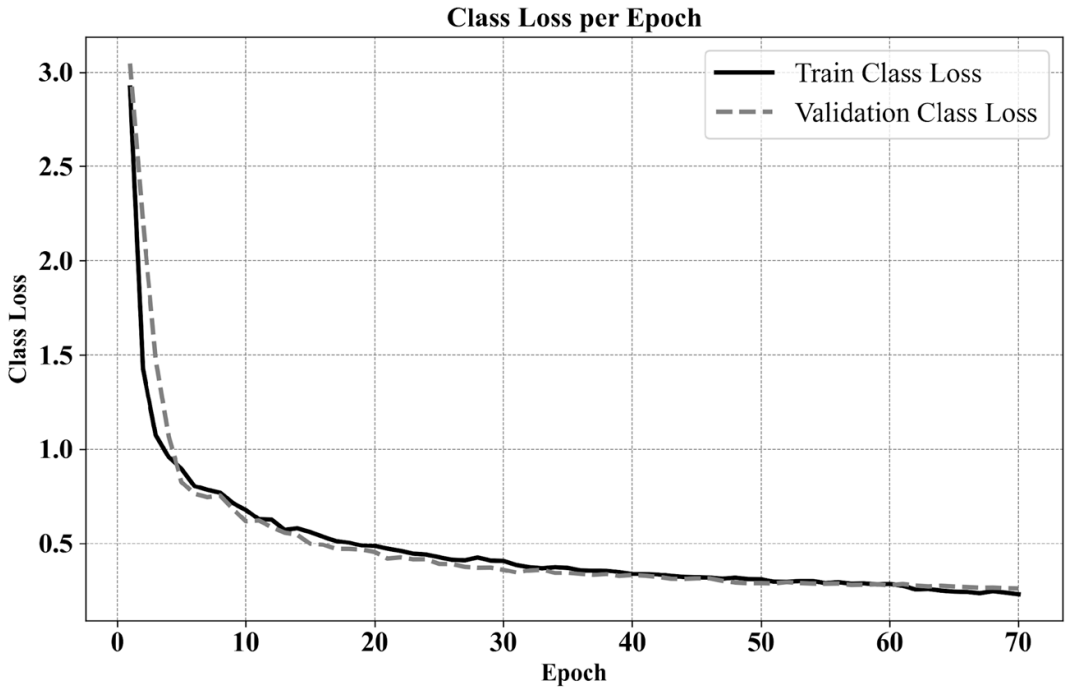

Launch Camera Manual
Launch Camera Auto
Solve Virtual
Solve Real
ScrambleVirtual
Scramble Real
Step Mode: ON
Display: ON
Use Arrow Keys to navigate.
1. Rotate Down Face 180 degrees in clockwise direction
2. Rotate Right Face 90 degrees in anti-clockwise direction
3. Rotate Front Face 90 degrees in clockwise direction
Total Moves Done:
LUD'D2R'F
2D Cube
Virtual Cube
Buttons For User Interaction
Total Moves = User Moves + Algorithm Moves
Solution by Algorithm

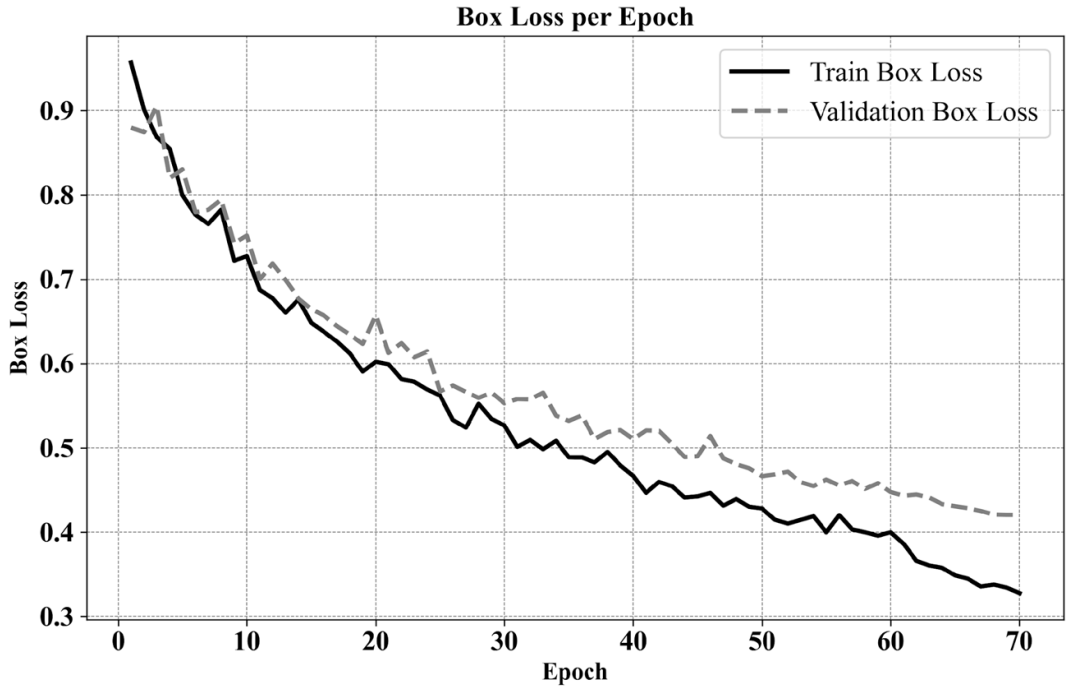
Result - [4] (GUI)



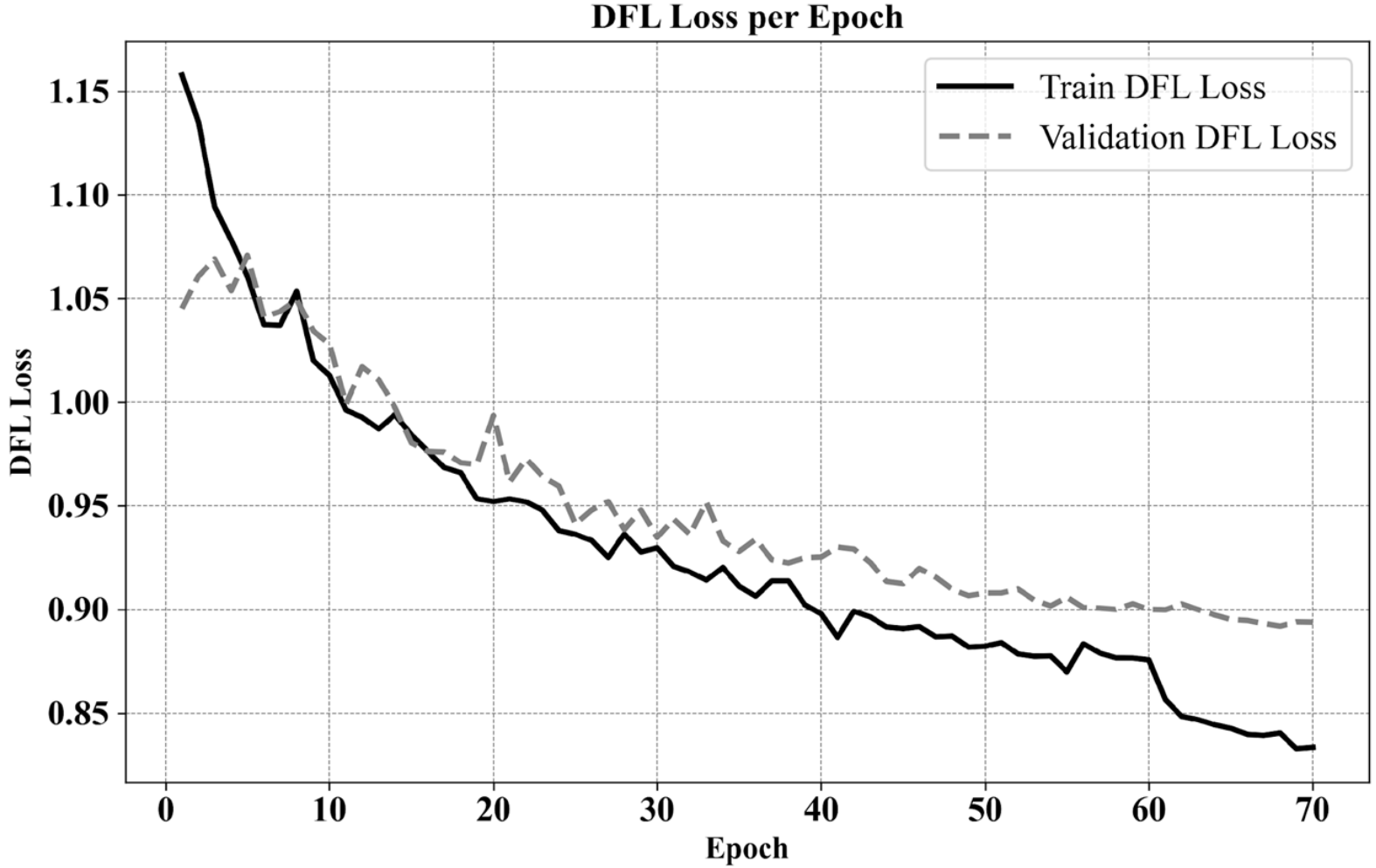
Button	Function
Launch Camera Manual	Launch Camera to capture Cube's state manually
Launch Camera Auto	Launch Camera to capture Cube's state automatically
Solve Virtual	Solve only the virtual cube
Solve Real	Solve virtual and real cube
Scramble Virtual	Scramble only the virtual cube
Scramble Real	Scramble virtual and real cube
Step Mode	Toggle between showing whole solution at once or step by step
Display	Toggle the visibility of Solution Pannel and Arrows

Result - [5]
(Class and Box Loss Per Epoch)

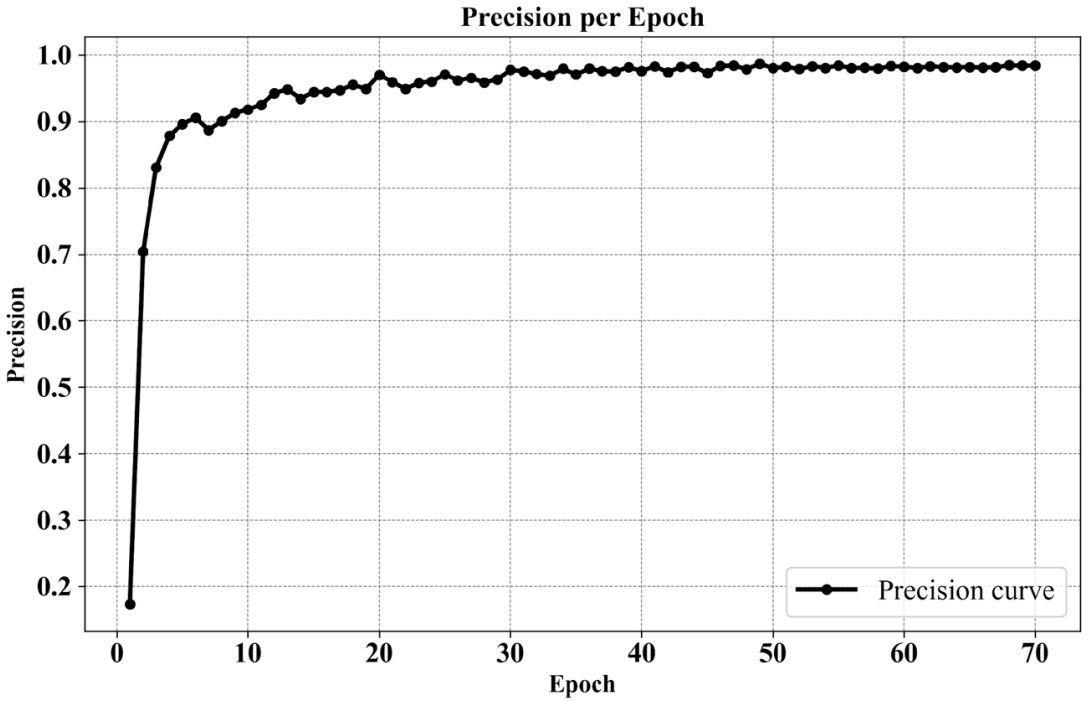


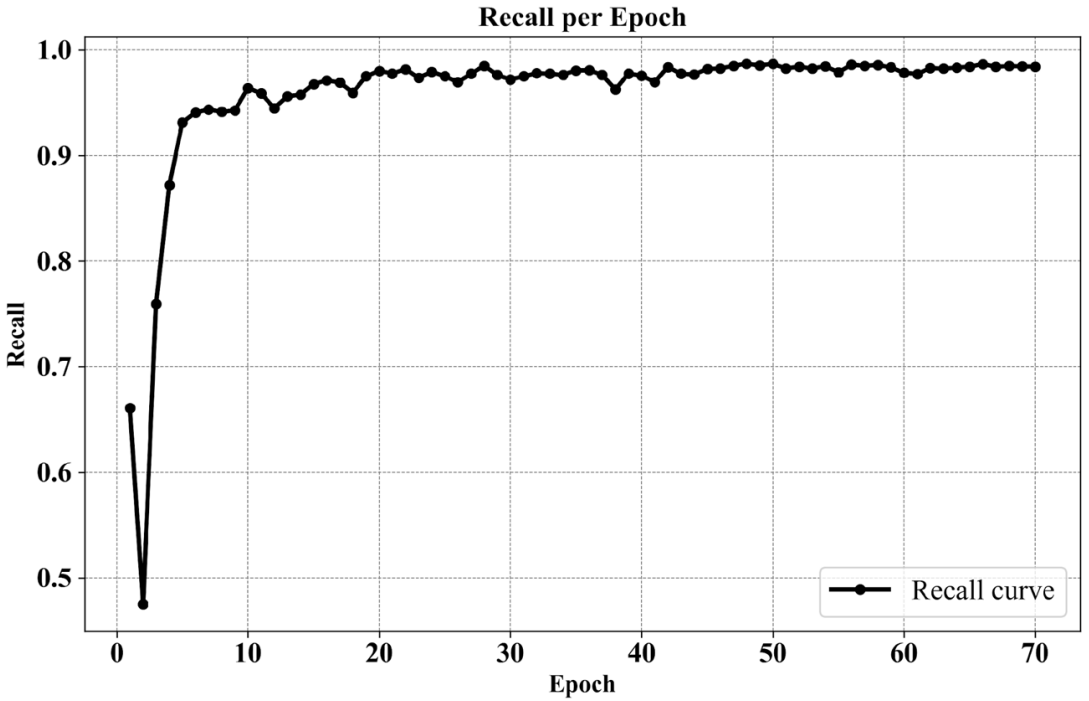


Result - [6]
(Distributive Focal Loss Per Epoch)

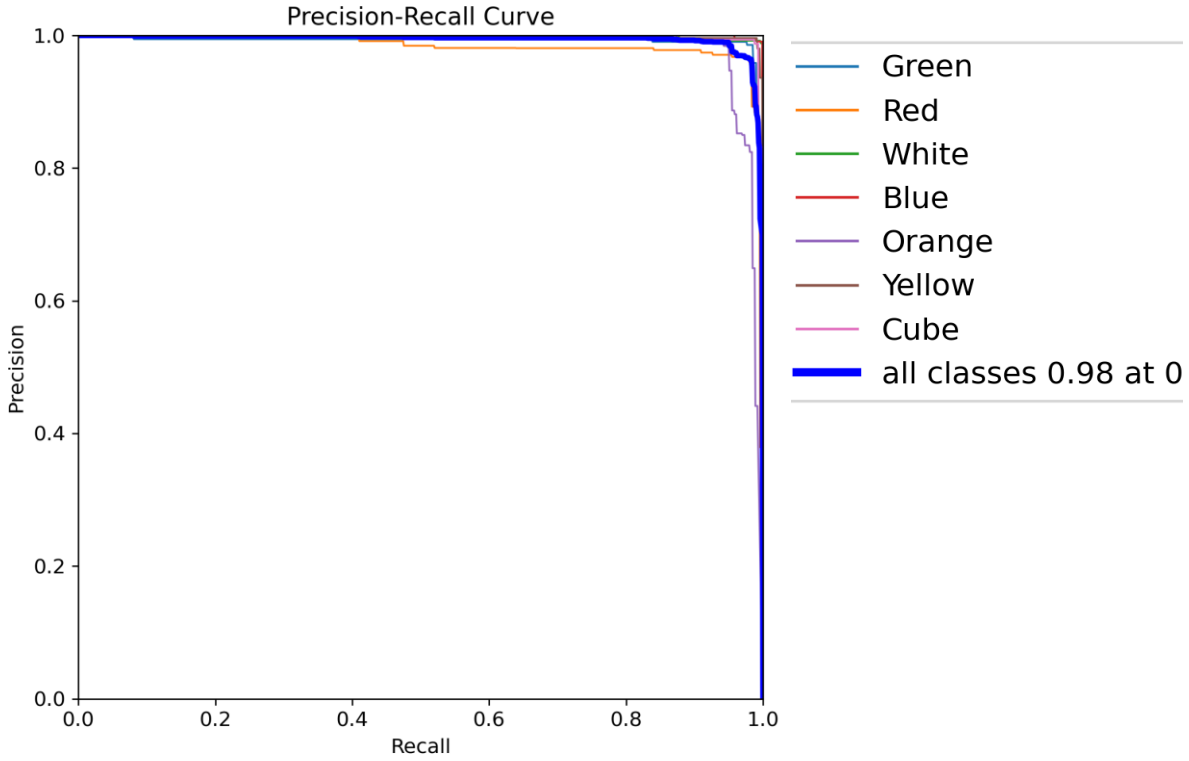


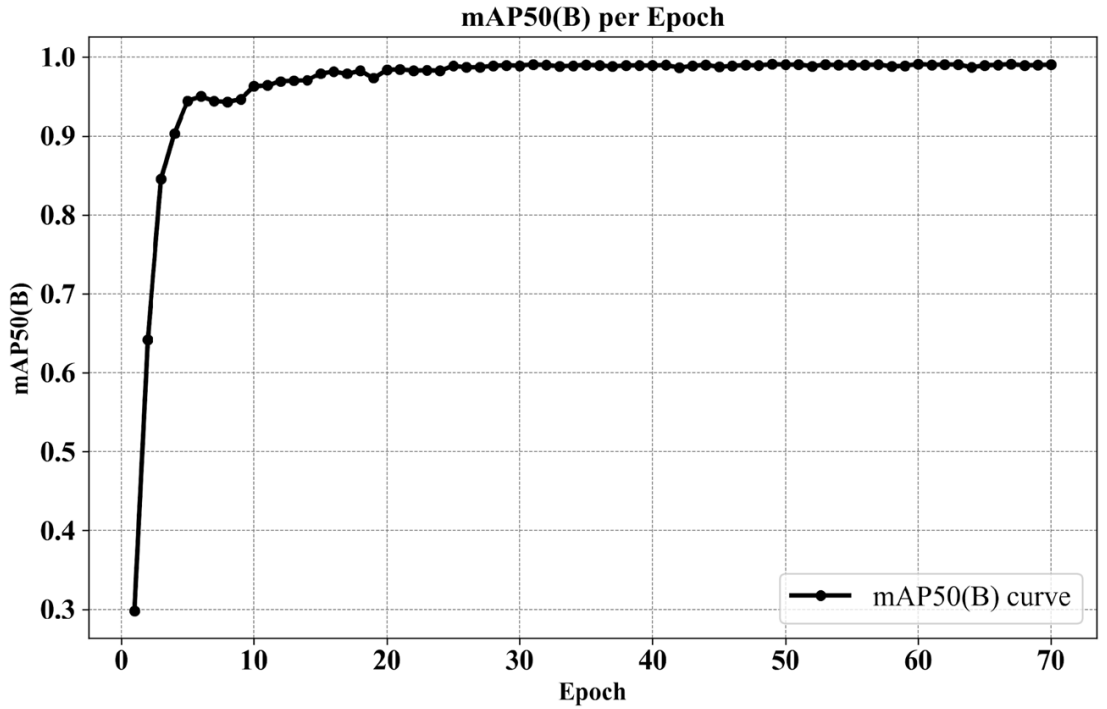
Result - [7]
(Precision and Recall Per Epoch)



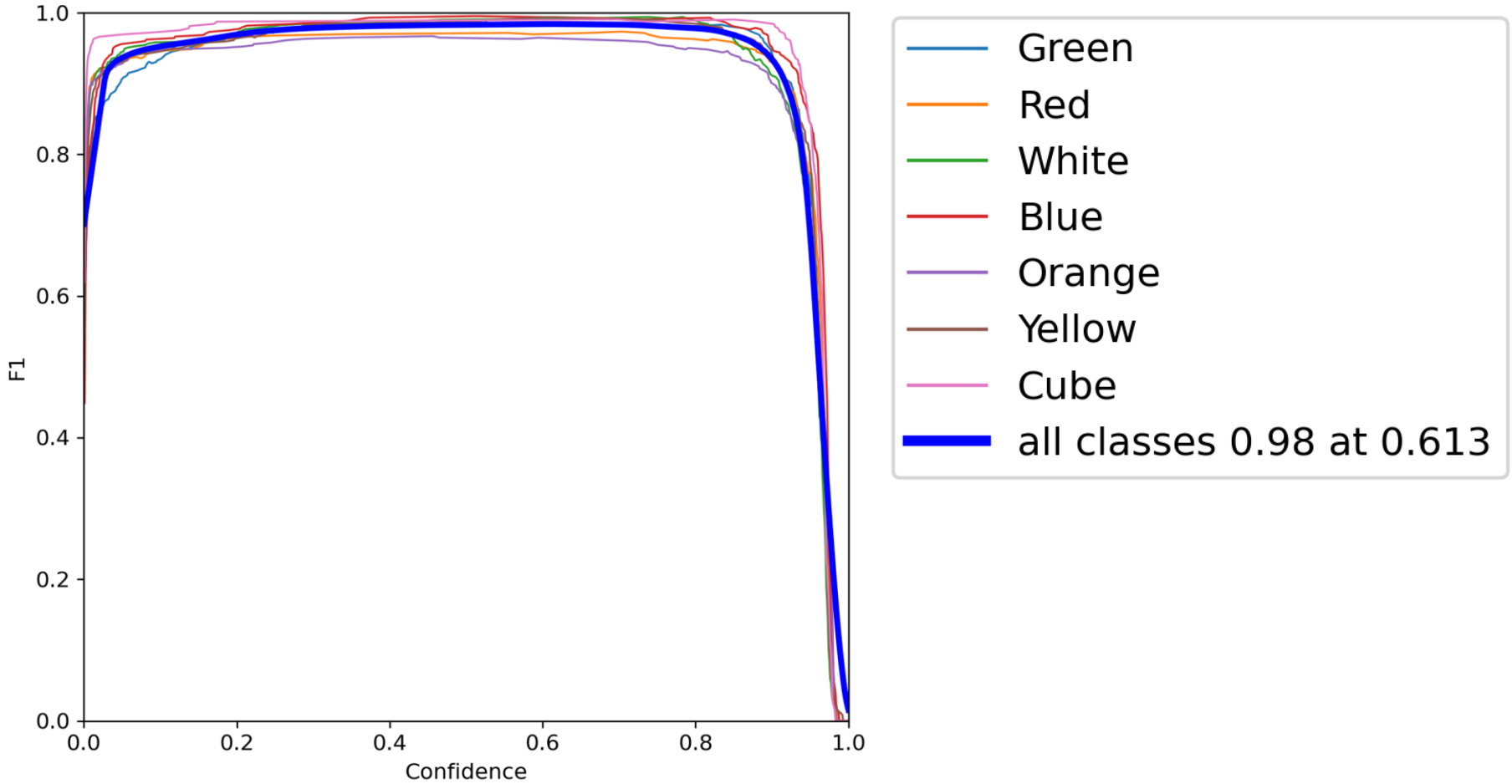


Result - [8]
(Precision-Recall and Mean Average Precision Curve)

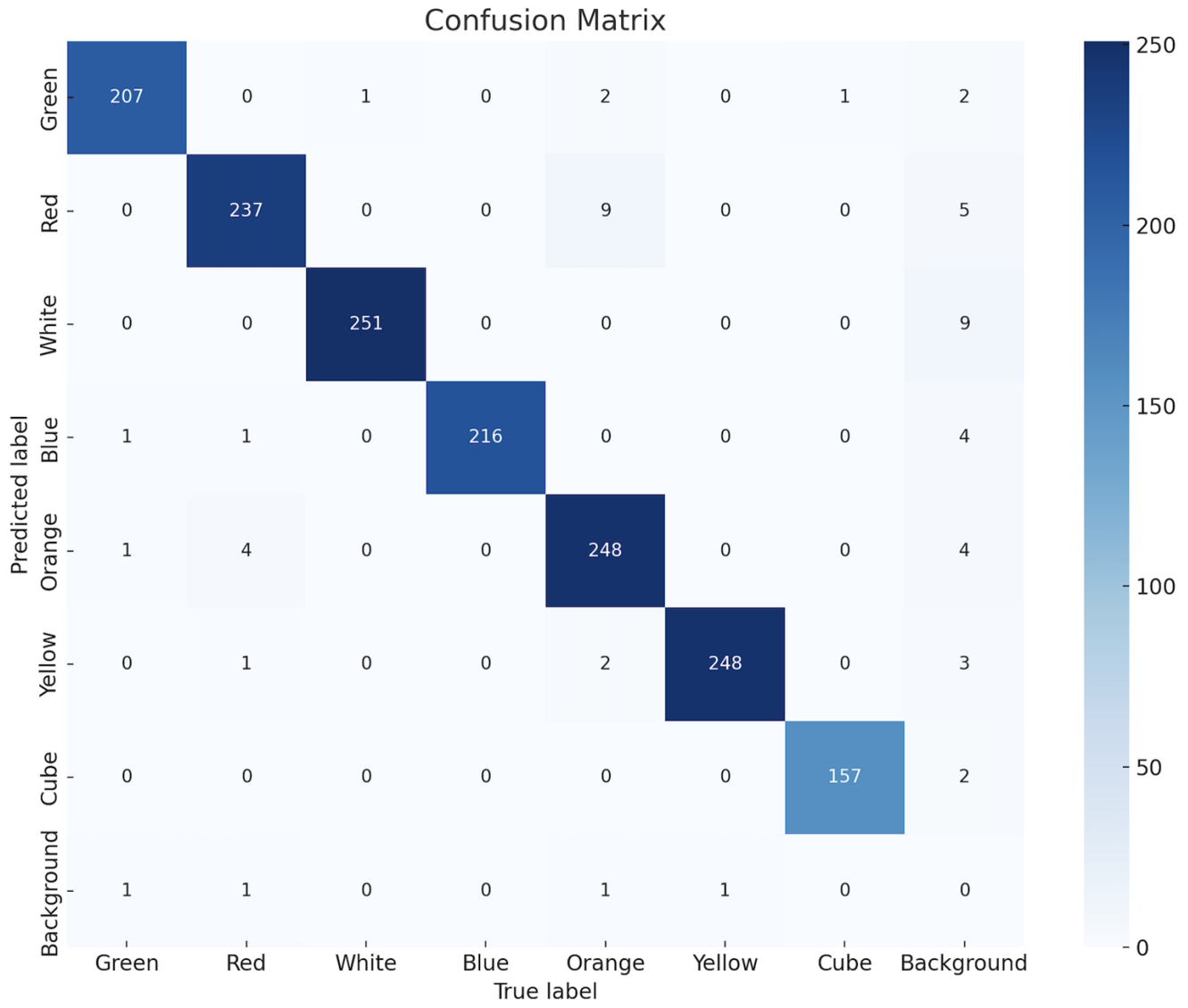




Result - [9]
(F1-Confidence Curve)



Result - [10] (Confusion Matrix)



Most Confused	Least Confused
Orange – 14x	Yellow - 1x
Red - 7x	White - 1x
Green - 3x	Blue - 0x

Result - [11]
(Time for Individual Moves)

Move Name	Time(s)
Flipping (f)	2.731
Rotating whole cube 90 degrees (r/R)	1.074
Rotating bottom layer 90 degrees clockwise direction (D)	2.028
Rotating bottom layer 90 degrees anticlockwise direction (d)	2.582
Rotating bottom layer 180 degrees (s)	3.319

Analysis - [1]

- Cube Detection Performance
  - Detection fails in drastic lighting conditions with confusion in colors red, yellow and orange due to them being close in color ranges.
  - Box loss isn't being smoothly reduced due to errors in box sizes in annotations
  - Class loss is being saturated probably due to some improper augmentations

Analysis - [2]

- Physical Model Performance
  - Average solving time: 2 minutes, 14 seconds.
  - Slow flipping due to torque requirement; flipper speed is a limiting factor.
  - Cube holder calibration offset reduces time to clockwise rotation compared to anticlockwise.
  - Rotating the bottom layer by 180° is not double the time for a 90° rotation due to the cube cover not needing to offset twice.
  - Fastest time for a 90° cube rotation; longest time for a 180° bottom layer rotation.

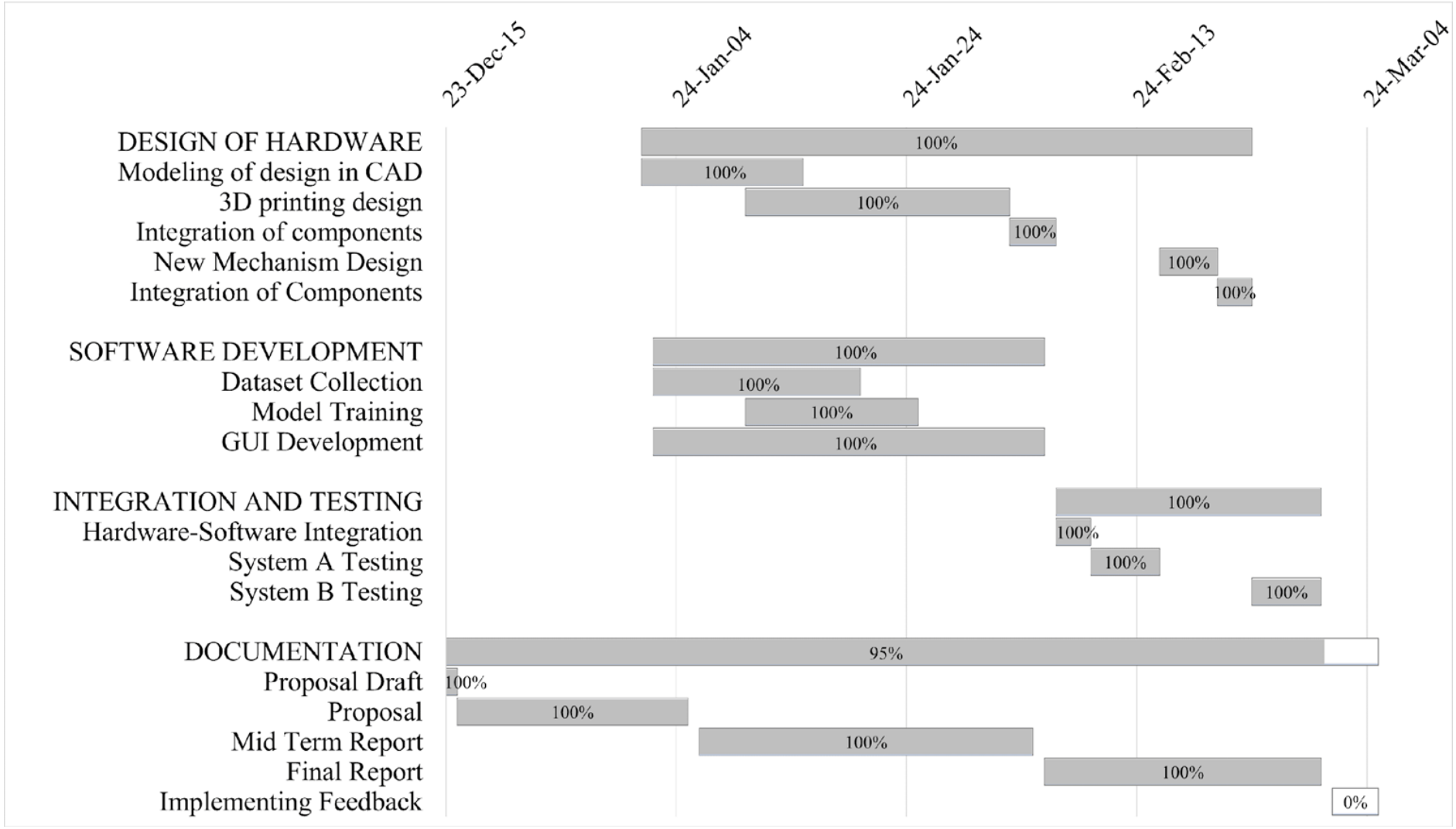
Future Enhancements

- Feedback System.
  - Auto calibration of starting position using magnetic encoders.
- Time Optimization.
  - Reduction of time for solving Rubik's cube by optimizing hardware.
- Platform Independence.
  - Make Software available in different Platform through web application.

Conclusion

- Successfully built a mechanical system that can solve a 3x3x3 Rubik's cube
- Successfully virtualized physical cube by capturing cube's faces
- Understood about building autonomous mechanical systems
- Developed skills in machine learning, software development
- Advanced the field intersecting robotics and machine learning

Timeline


23-Dec-15
24-Jan-04
24-Jan-24
24-Feb-13
24-Mar-04
DESIGN OF HARDWARE
Modeling of design in CAD
3D printing design
Integration of components
New Mechanism Design
Integration of Components
SOFTWARE DEVELOPMENT
Dataset Collection
Model Training
GUI Development
INTEGRATION AND TESTING
Hardware-Software Integration
System A Testing
System B Testing
DOCUMENTATION
Proposal Draft
Proposal
Mid Term Report
Final Report
Implementing Feedback
100%
95%
0%

Project Budget

Resource	Quantity	Cost	Total
3D Printing Filament	1 KG	Rs. 3,000 per kg	Rs. 3,000
3D Printing Cost	25 hours	Rs. 70 per hour	Rs. 1750
Stepper Motor (STH-39H112-06)	3 units	Rs. 1600 per unit	Rs. 4800
Motor Driver (A4988 Stepper Driver Module)	3 units	Rs. 250 per unit	Rs. 750
Rubik's Cube	1 unit	Rs. 500 per unit	Rs. 500
Arduino UNO Microcontroller	1	Rs. 1500 per unit	Rs. 1500
Power Adapter	3	Rs. 300 per unit	Rs. 900
Grand Total			**Rs. 13,700**

References - [1]

[1] S. Lu, M. Huang, and F. Kong, “The Design of a Rubik’s Cube Robot,” Advanced Materials Research, vol. 709, pp. 432–435, 06 2013.

[2] V. Dan, G. Harja and I. Naşcu, “Advanced Rubik's Cube Algorithmic Solver,” 2021 7th International Conference on Automation, Robotics, and Applications (ICARA), Prague, Czech Republic, 2021, pp. 90-94, doi: 10.1109/ICARA51699.2021.9376564.

[3] E. S. Toshniwal and Y. Golhar, “Rubik’s Cube Solver: A Review”, 2019 9th International Conference on Emerging Trends in Engineering and Technology - Signal and Information Processing (ICETET-SIP-19), Nagpur, India, 2019, pp. 1-5, doi: 10.1109/ICETET-SIP-1946815.2019.9092272.

[4] S. McAleer, F. Agostinelli, A. Shmakov, and P. Baldi, “Solving the Rubik’s Cube Without Human Knowledge,” arXiv.org, May 18, 2018. https://arxiv.org/abs/1805.07470 (accessed Dec. 16, 2023).

References - [2]

[5] F. Agostinelli, S. McAleer, A. Shmakov, and P. Baldi, “Solving the Rubik’s cube with deep reinforcement learning and search,” Nature Machine Intelligence, vol. 1, no. 8, pp. 356–363, Aug. 2019.

[6] M. E. Chasmai, “CubeTR: Learning to Solve The Rubiks Cube Using Transformers,” arXiv.org, Nov. 11, 2021. https://arxiv.org/abs/2111.06036 (accessed Dec. 16, 2023).

[7] kkoomen, “GitHub - kkoomen/qbr: A webcam-based 3x3x3 rubik’s cube solver written in Python 3 and OpenCV.,” GitHub. https://github.com/kkoomen/qbr (accessed Jan. 02, 2024).

[8] T. Rokicki, H. Koceimba, M. Davidson, and J. Dethridge, “God’s Number is 20.,” https://www.cube20.org/ (accessed Dec. 13, 2023).

Thank You