PES
Institute of Technology

Rubik's Cube Problem Solver

By:

Sindhuri KN	1PI08IS 097
Varsha Abhinandan	1PI08IS 117
Vedashruti Pandiyan	1PI08IS119

Agenda

- Project Overview – Problem Statement
- Motivation
- Project Overview
- Requirements
- System Design
- Rubik’s cube solving algorit
- Project Phase 1
- Project Phase 2
- References

Motivation

- Intriguing puzzle
- Difficulty faced in solving the Rubik’s cube
- Unconventional application of image processing and computer vision techniques
- Domain in which linear algebra

Project Overview

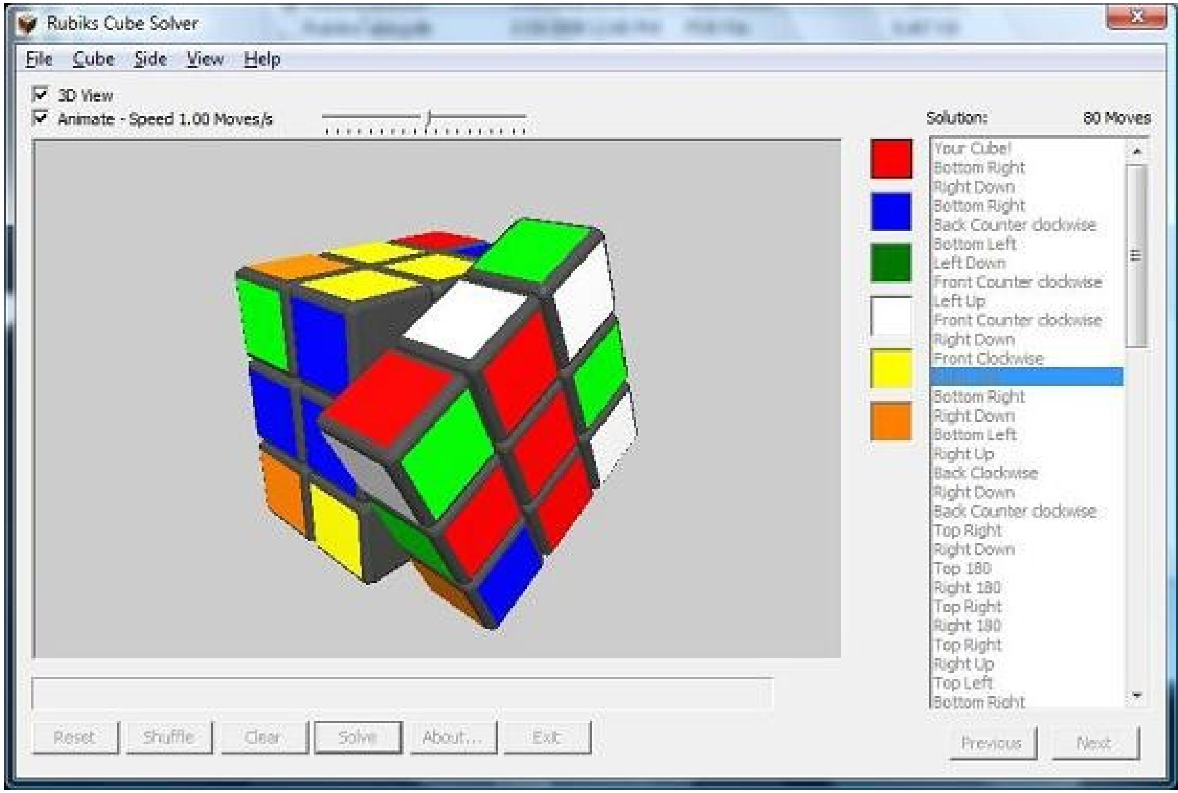
Problem statement:

- To develop an application that allows the user to let the system scan an unsolved 3×3×3 Rubik's Cube and determine the next steps to be taken to solve it.

1. Identifying the current configuration of the Rubik's cube.
2. Apply puzzle solving algorithms.

Existing System

- Take input from user **manually** about which colors are on which faces.
- **Tutorials**:
  - Explain Rubik's cube solving algorithm
  - Generate random puzzles and ask user to solve



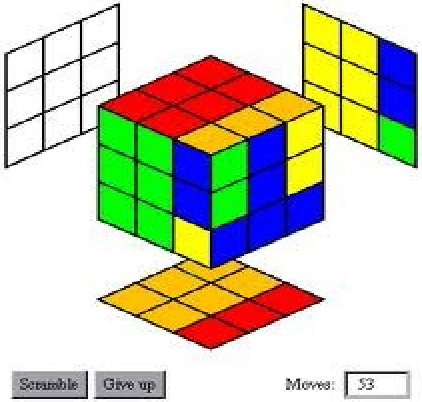
☑ 3D View
☑ Animate - Speed 1.00 Moves/s

Solution: 80 Moves

```
Your Cube!
Bottom Right
Right Down
Bottom Right
Back Counter clockwise
Bottom Left
Left Down
Front Counter clockwise
Left Up
Front Counter clockwise
Right Down
Front Clockwise

Bottom Right
Right Down
Bottom Left
Right Up
Back Clockwise
Right Down
Back Counter clockwise
Top Right
Right Down
Top 180
Right 180
Top Right
Right 180
Top Right
Right Up
Top Left
Bottom Right
```

Proposed System



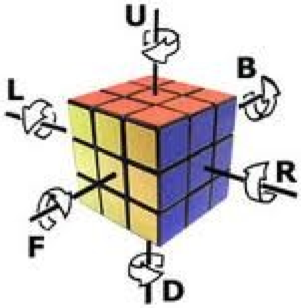
Unsolved Rubik’s cube

Camera

System

Identify current state of the Rubik’s cube

Apply Rubik’s cube solving algorithms



Display steps to solve the Rubik’s cube

Requirements

- Hardware requirements:
  - Rubik's cube
  - System with camera (webcam is good enough)
  - RAM - 1GB
  - Processor speed – 2.4GHz

- Software requirements:
  - Visual Studio 2008 Express Edition
  - OpenCV 2.3.0 - computer vision library
  - OpenGL Version 2.1 – computer graphics library

System Design

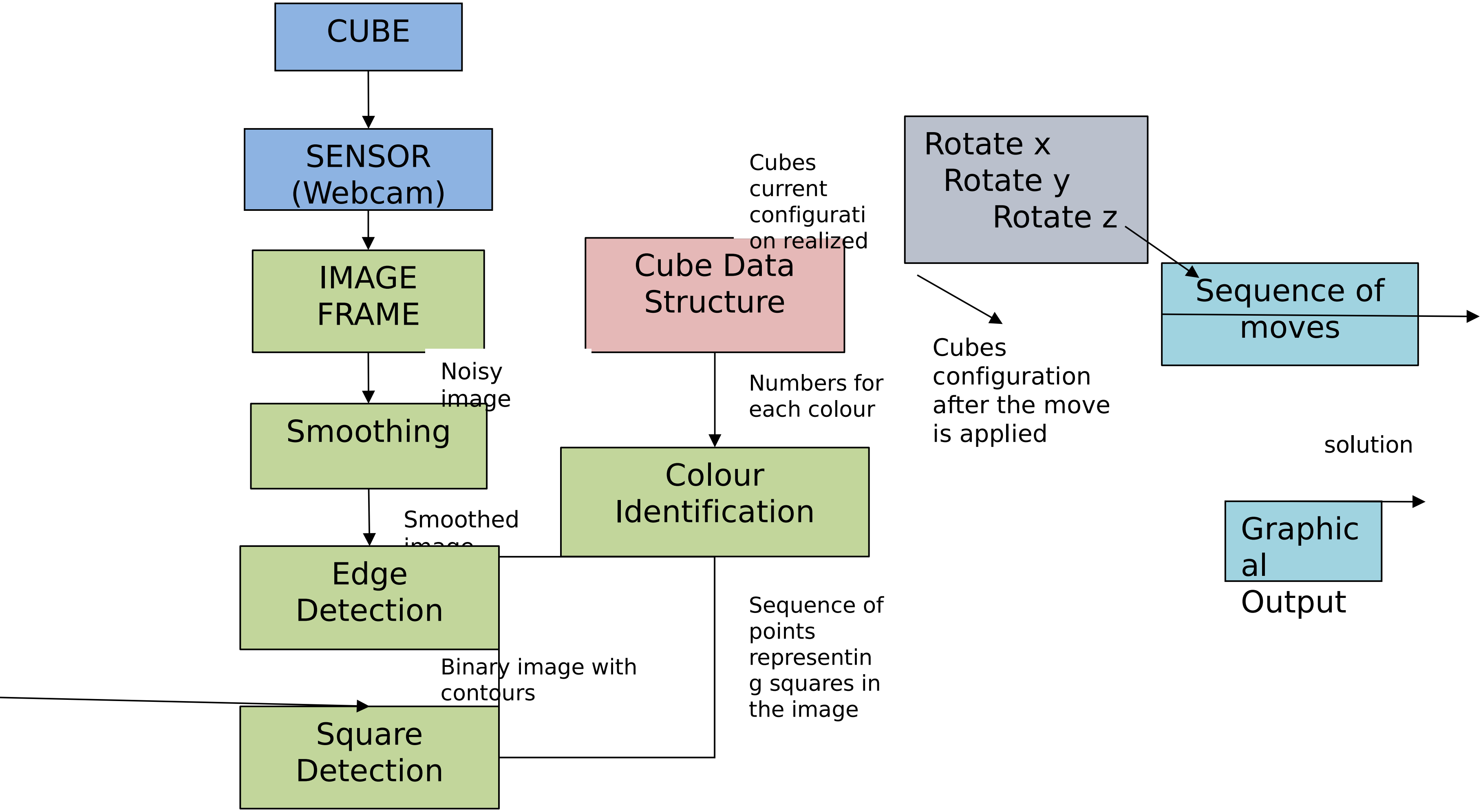


Figure : Data Flow Diagram

OpenCV

- Open source **computer vision library** written in C and C++

- Runs under Linux, Windows and Mac OS X

- Designed for **computational efficiency** with strong focus on real-time applications

- Optimized in C and can take advantage of **multicore processors**

- Contains over 500 functions that span

Canny Edge Detection

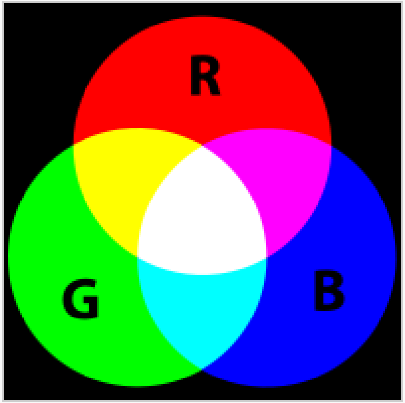
- **Noise Removal**
- **Finding gradients**: The edges should be marked where the gradients of the image has large magnitudes.
- **Non-maximum suppression:** Only local maxima should be marked as edges.
- **Double thresholding:** Potential edges are determined by thresholding.
- **Edge tracking by hysteresis:** Final edges are determined by suppressing all edges that are not connected to a very certain (strong) edge.

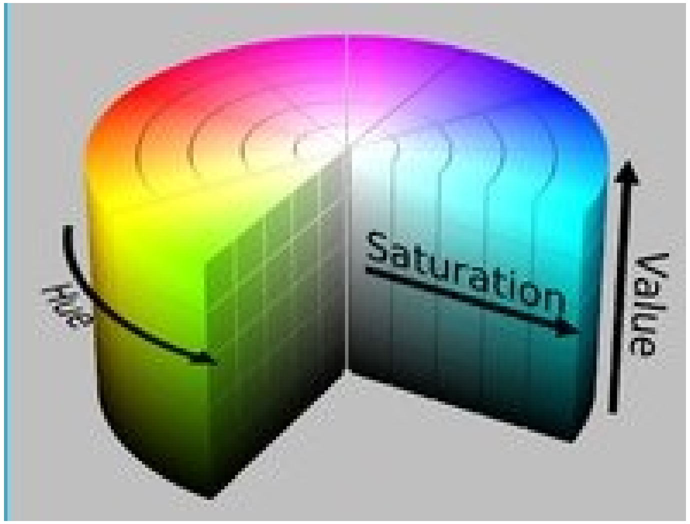
Square Identification

- Edge pixels obtained – **assemble into contours**
- **Contour** – list of points that represents curve in an image
  - cvFindContours()
- Approximate contours to polygon
- Check if the polygon can be a required square:
  - 4 points
  - Convexity
  - Angle constraint

Colour Identification

- Clustering based on **hue** of intersection points – group all centre pixels into 6 clusters (6 colours)
- Colour spaces:





RGB colour format can represent any standard colour or brightness using a combination of Red, Green and Blue components

HSV colour format:
Hue of a colour remains the same irrespective of the brightness

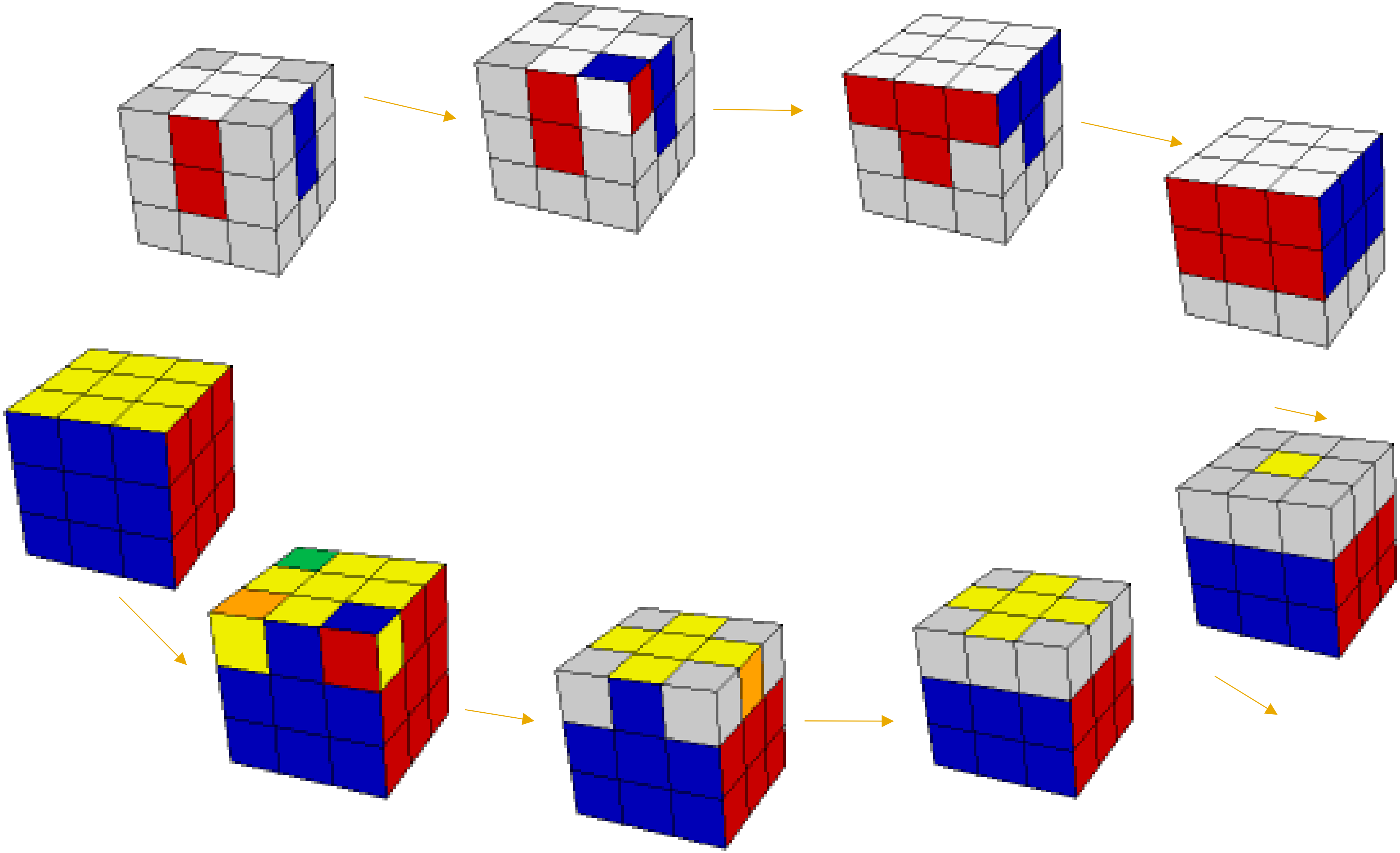
Representation of Rubik’s Cube

- The **face** of each cube
  - 3x3 array
  - Pointers to the adjacent faces

```
struct face
{
      int arr[3][3];
      struct face *u;
      struct face *l;
      struct face *r;
      struct face *d;
};
```

- **Sticker colours**
  - Numbers from 1-6

Rubik's Cube Solving Algorithm



Project Phase 1

3 independent modules have been developed:

1. Capturing input frames

2. Detection of cubie colors

3. 1st step in solving the Rubik’s cube: formation of cross on first layer

Project Phase 2

- Solving the Rubik’s cube completely
- Show graphical solution using OpenGL
- Improvising the Square detection and integrating it with colour detection

Open GL

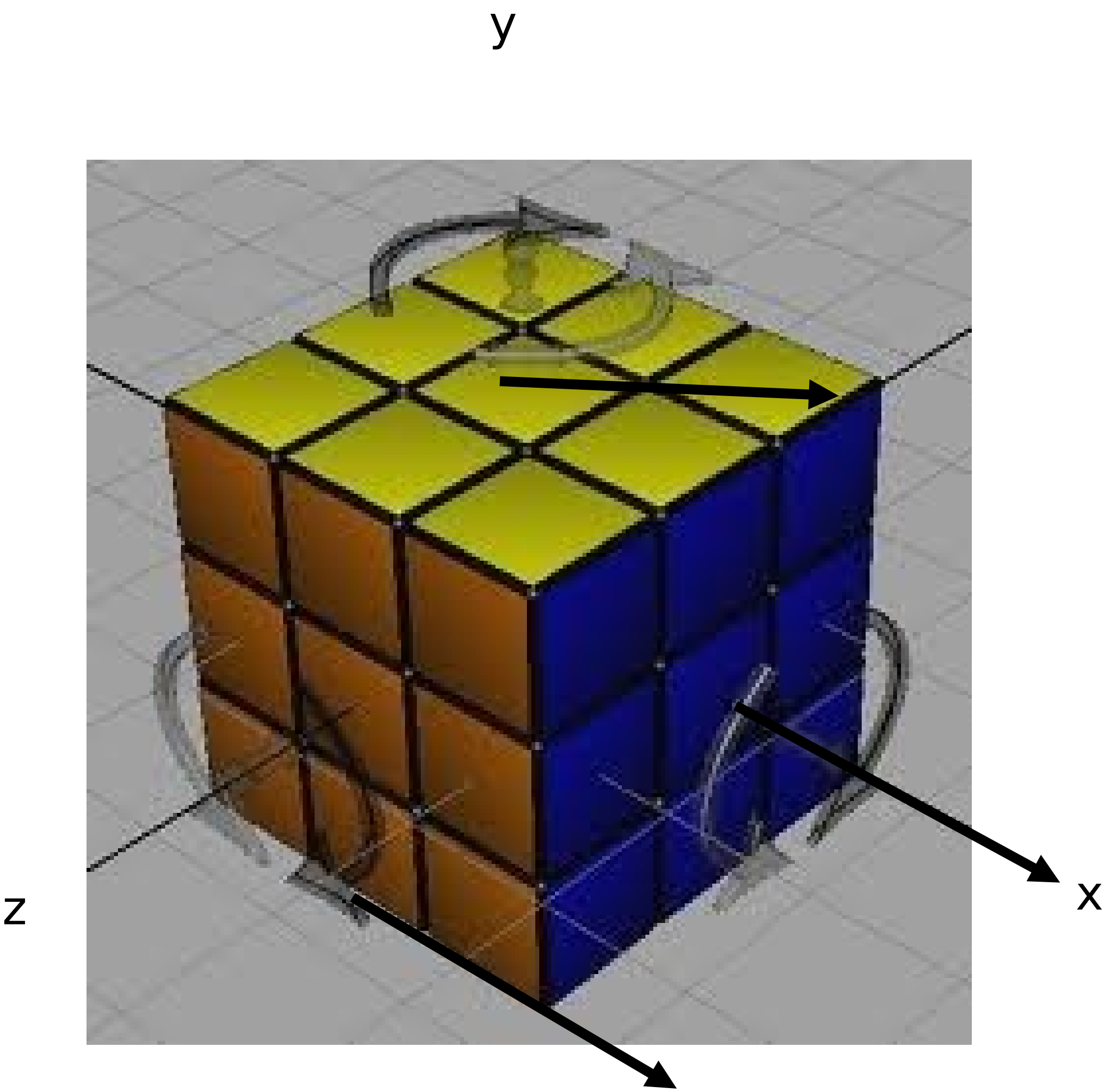
- hardware.

- It is a low-level graphics library specification consisting of a small set of geometric primitives - points, lines, polygons, images, and bitmaps.

- The OpenGL Utility Library (GLU) provides most of the modeling features to draw various surfaces and curves .

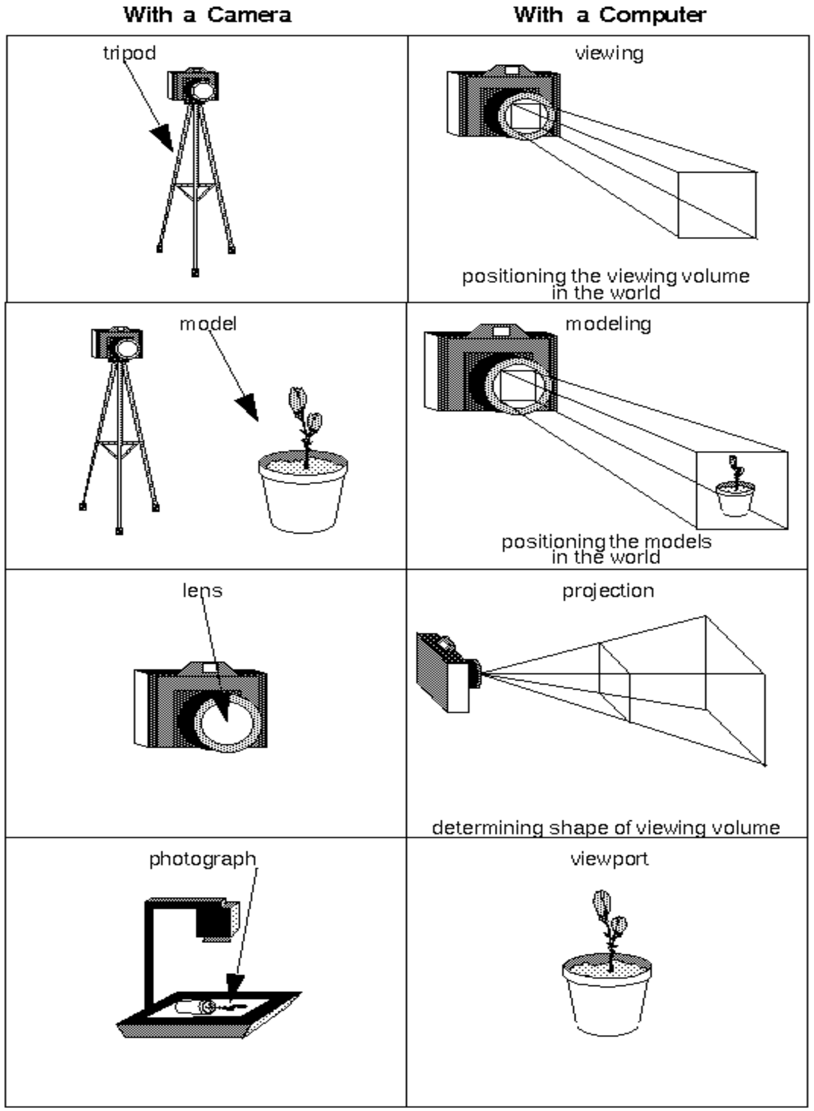
- The major graphics operations which OpenGL performs to render an image on the screen are:
  - Construct shapes from geometric primitives
  - Arrange the objects in three-dimensional space and select the desired vantage point for viewing.
  - Calculate the color of all the objects

3D Model of Rubik's Cube



The Camera Analogy

- Viewing transformation
- Modeling transformation
- Projection transformation
- Viewport transformation
- Drawing the scene


With a Camera
With a Computer
tripod
viewing
positioning the viewing volume in the world
model
modeling
positioning the models in the world
lens
projection
determining shape of viewing volume
photograph
viewport

ANIMATEABLE

DEMONSTRATION

Future Enhancements

- Advanced Solution to solve the Rubik’s Cube in fewer steps
- Reading three faces of the at once
- Improve the system to work with non-std cubes
- Improve user interface
- Code optimization and integration into a cube solving Robot
- Providing a better user Interface

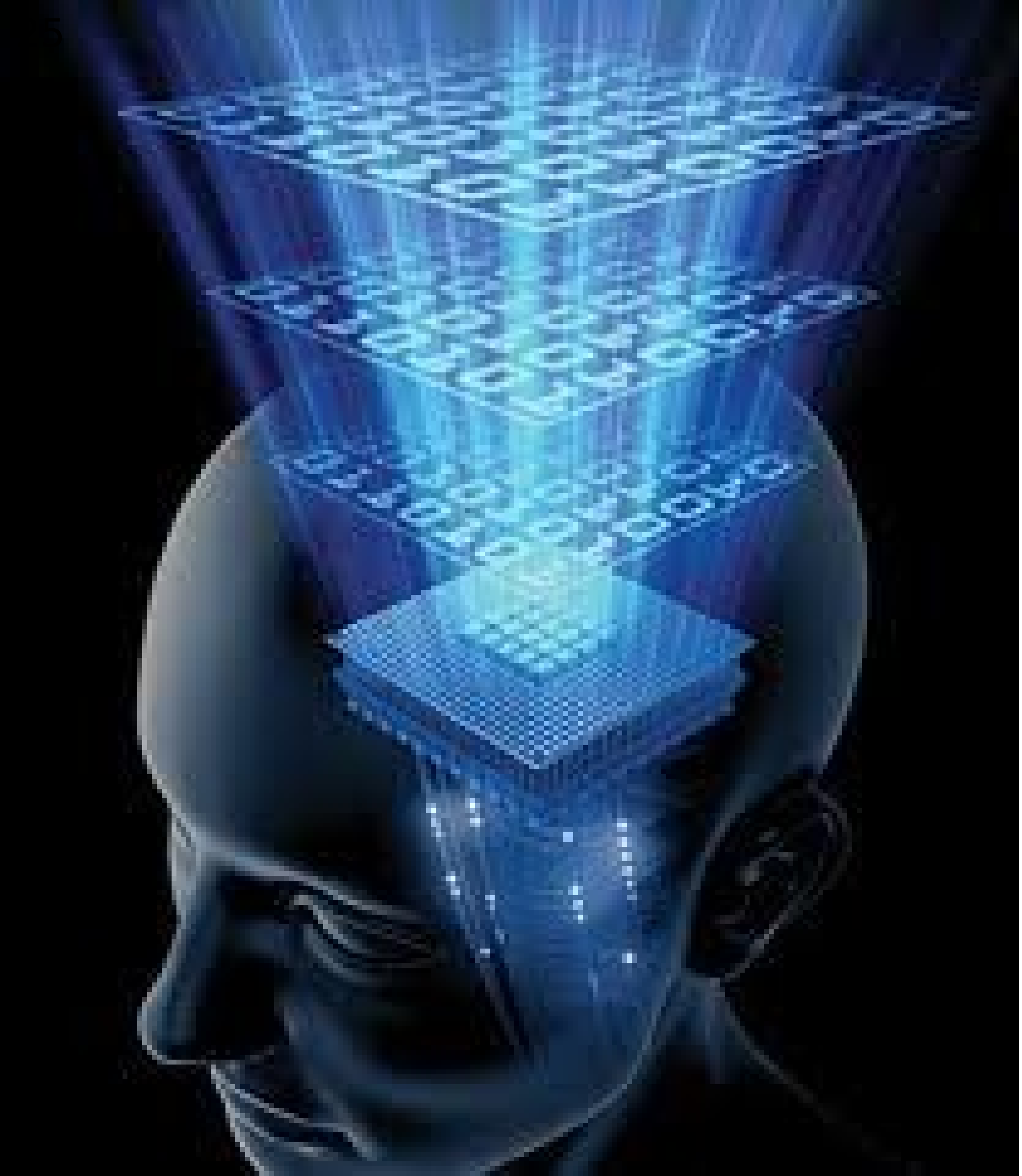
Applications

- Learning Aid for Rubik’s cube enthusiasts and beginners

- Kernel program to evaluate system performance

References

1. Gary Bradski and Adrian Kaehler, Learning OpenCV, O’Reilly, First Edition
2. Richard Szeliski, Computer Vision: Algorithms and Applications, Springer.
3. OpenGL programming guide, the official guide to learning openGL, Version 2.1 (6th edition, pearson publication) by dave shreiner, mason woo, jackie neider, tom davis..
4. http://www.cs.cmu.edu/~cil/vision.html
5. http://en.wikipedia.org/wiki/Computer_vision
6. http://en.wikipedia.org/wiki/Segmentation_%28image_processing%29
7. http://www.discover.uottawa.ca/~qchen/my_presentations/A%20Basic%20Introduction%20to%20OpenCV%20for%20Image%20Processing.pdf
8. http://en.wikipedia.org/wiki/Rubik%27s_Cube
9. http://homepages.inf.ed.ac.uk/rbf/HIPR2/canny.html
10. http://homepages.inf.ed.ac.uk/rbf/HIPR2/threshld.html



Thank You