## Table of Contents



# CS 180: Poor Man's Augmented Reality

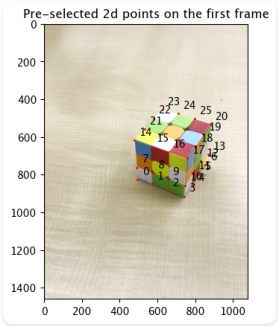

**By Ethan Jagoda**

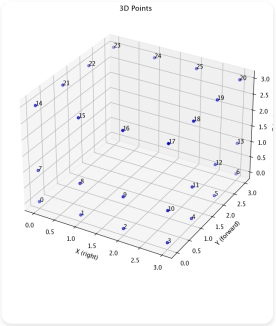

## Introduction

This project implements a simplified augmented reality system. We capture a video of a scene with a known object, track keypoints with known 3D coordinates, estimate the camera's projection matrix for each frame, and insert a synthetic cube into the scene. The cube should appear to remain fixed in 3D space despite camera motion.

## Setup

We start by placing a box, in this case a rubik's cube, with a known pattern and measuring key distances so we can define a consistent 3D coordinate system. Fortunately, the cube has a grid pattern on each face, so we can easily use the existing grid to define a 3D coordinate system. These points provide the anchor for camera calibration throughout the video.

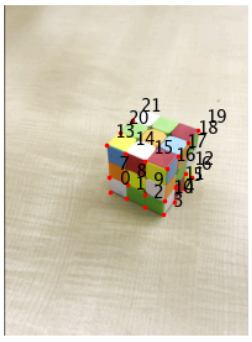


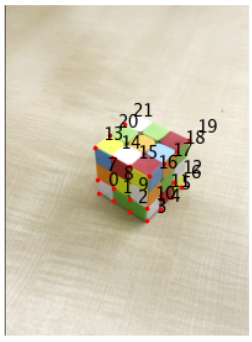



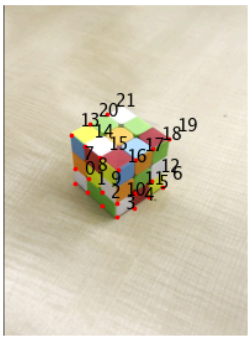

## Tracking Points

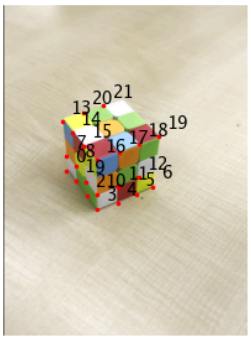
Now that we have the initial anchor points, we need to track the 2D positions of these marked points across all frames of the video. Using OpenCV's MedianFlow tracker, I initialized small bounding boxes around each point in the first frame and update them frame-by-frame. This yields a set of stable 2D correspondences for every frame, matched to fixed 3D points.

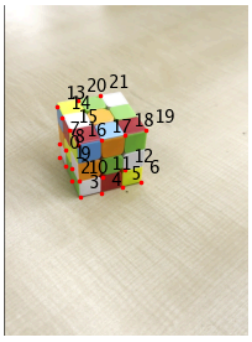
Note: Because of the pattern in the woodgrain background, a few of the points were not tracked well. This was easy to fix by manually removing them manually.











## Camera Calibration

With the 3D-2D correspondences for each frame, we solve a least-squares problem to compute the camera projection matrix. This matrix maps 3D points in the known coordinate system to 2D image coordinates for that frame.

Recomputing it for each frame accounts for camera movement.

The general camera projection matrix P:

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}$$

It maps 3D world coordinates:

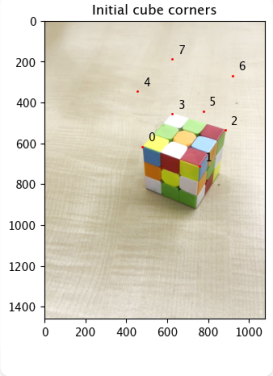
$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

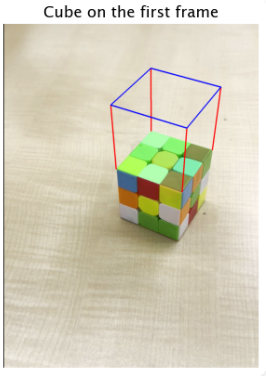
to 2D image coordinates:

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \mathbf{P} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

## Rendering the Cube

After obtaining the projection matrix, we project synthetic 3D points defining a cube onto the 2D image. The cube is drawn on each frame, appearing as if it is anchored in the real scene. We then stitch these annotated frames back into a video to produce the augmented reality effect.

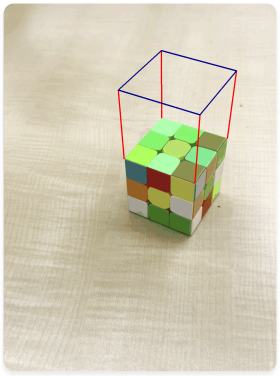



## Results

The final output shows the input video with the overlaid synthetic cube. The cube remains stable as the camera moves, demonstrating proper camera calibration and point tracking.

Note: This is a gif, let me know if this is not working.



## Table of Contents