AUGMENTED REALITY

3D Object Insertion in REAL-TIME Camera Using OpenCV

Presented by Zhihao Cao

INTRODUCTION

- Augmented Reality

  To displays the combining view of computer-generated input overlaying the reality view from camera.

MICROSOFT HoLo-Lens (2:20)

PROBLEM STATEMENT

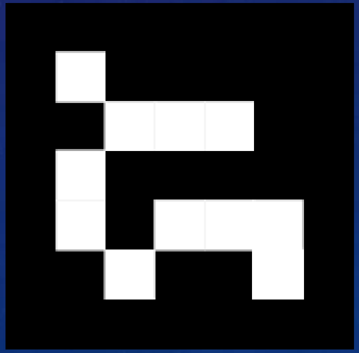
- To achieve the goal which displays an object in the specific positon of the camera view, a marker is needed to help the program to locate the position.
- Many techniques are designed to recognize the marker, but the accuracy of recognition is never enough.
- The main reasons are:
  - Marker color changes due to the environment (illuminant)
  - Marker shape changes due to the angle and position of camera
  - Part of Marker is blocked by some objects
  - Marker moving too fast, thus camera unable to capture a clear input.

PREVIOUS WORK

- Various marker detection techniques are proposed. They are basically in these two categories.
- 1. Measure the degree of correlation of a pattern (found inside an image) with known patterns (Feature detection with descriptor)
- 2. Use digital methods: read a binary code from the marker. The code stores a non redundant ID for identification.
- Correlation technique usually performs less robust than the digital code method.

PRESENTATION OF SOLUTION

- In this project, a synthetic square marker method is applied.
- The marker is composed by a wide black border and a inner binary matrix which determines its identifier (id) and original orientation (without asymmetry).

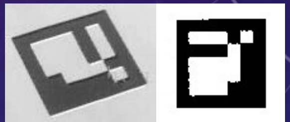


Marker Detection And Recognition

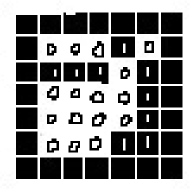
- 1. Convert camera frame to gray image.
- 2. Apply adaptive thresholding to the gray image, so the output is a segmented binary image.
- 3. Apply morphologyEx() - open operation to reduce the noise.
- 4. Apply findContours to extract all possible markers.
- 5. Filter out the non-markers by applying criteria: only four corners, convex.
- 6. Only real markers remain. (Detection complete)

Marker Detection And Recognition

- 1. Apply getPerspectiveTransform() and warpPerspective() to obtain marker’s the canoni



- 2. Apply Otsu's thresholding to obtain the binarization image.
- 3. Build the bit matrix by categorizing the value of each cell inside the maker.
- 4. Decode the bit matrix to find the right orientation and rotate.
- 5. In the right orientation, decode the mark ID.
- 6. Mark recognition complete.



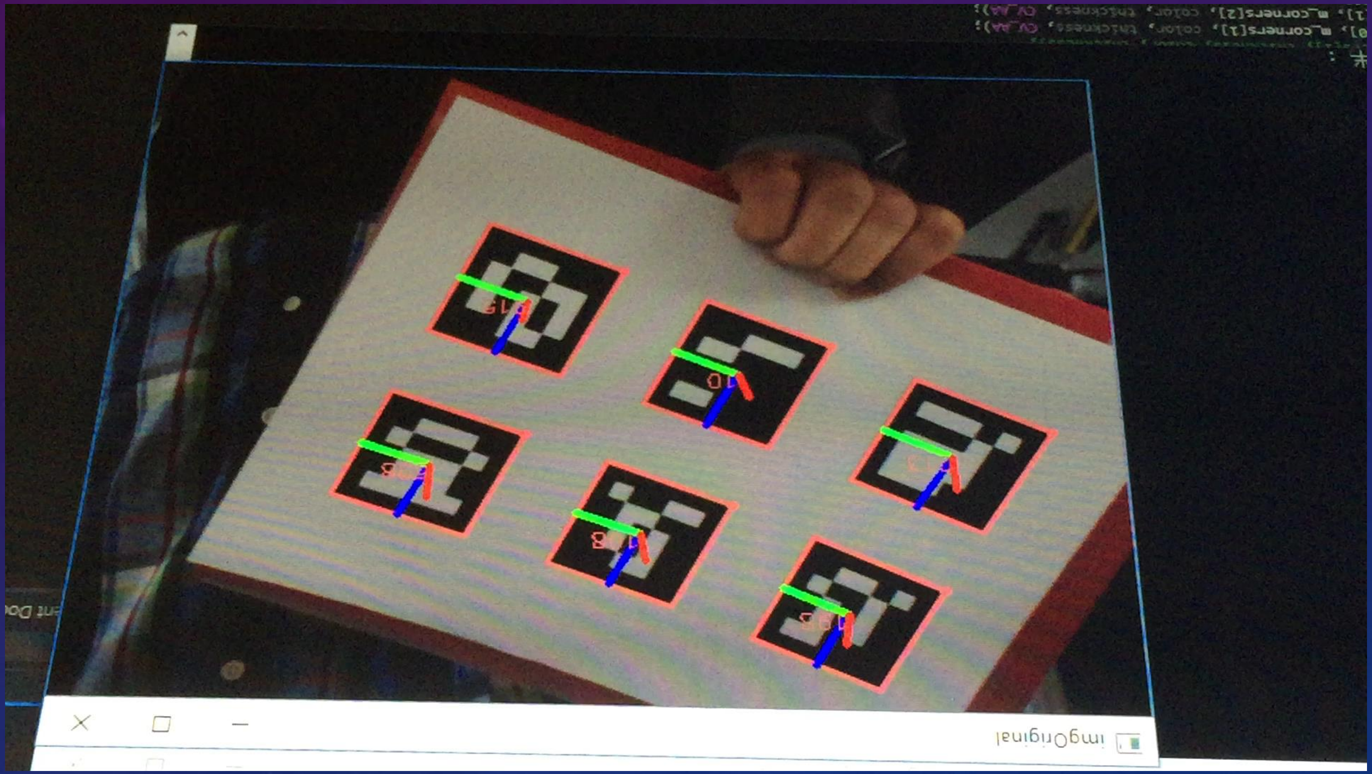
1	1	1	0	1
0	0	0	1	0
1	1	1	1	0
1	1	1	1	0
1	1	1	0	0

5x5

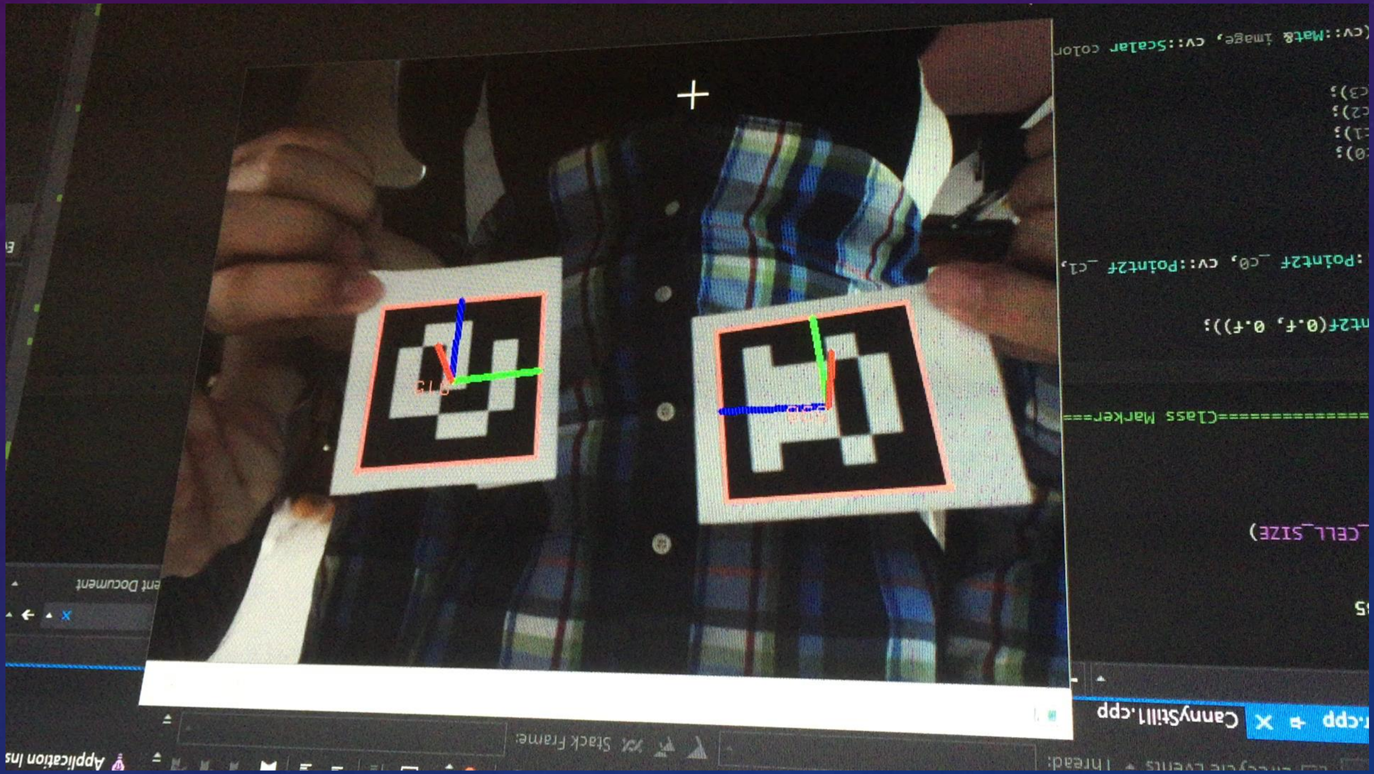
Calculate Camera Position

- 1. Apply calibrateCamera() to obtain the intrinsic parameters.
- 2. Apply solvePnP() to obtain the extrinsic parameters.
- 3. Use these parameters to draw axis on each marker.
- 4. Using OpenGL to draw 3D model (or animating model) into the marker position (still working on it).

Presentation Of Experiments And Results



Presentation Of Experiments And Results



Future Update

- 1. Import 3D model using OpenGL.
- 2. Apply more advanced detection technique to recognize complex objects (Descriptor).
- 3. Write an Android version of this program and run on a smartphone.

THANK YOU!