Augmented Reality Checkers
(English Draughts)

Nik Cimino
Introduction

• **Purpose**
  - New user interfaces are a growing market
  - Games make money, and are fun!

• **Goals**
  - Use computer vision to interface the physical world to the digital world
  - Use OpenCV (C++ Library) to build the aforementioned interface

• **Assumptions**
  - Checkerboard needs to be close to camera
  - Surroundings should have limited numbers of edges
Research

• Color based edge detection – Piece identification
  o Similar colors differentiation can be optimized using color based edge detection (1)
  o The checkers game pieces are often similar in to the checkerboard, in color

• Checkerboard detection
  o Hough lines can be grouped with voting (similar to RANSAC) for pose estimation (2)
  o Pose estimation is essential for deterministic results

Color Edge Detection

- Research suggests color based edge detection
- In practice, grayscale yielded the optimal result
Color Edge Detection (cont.)

Blue

Grayscale

Greed

Red
Checkerboard Identification

- Attempted similar angles approach to find corners
- In practice, the outer limits were not the corners
- This approach did not work
Checkerboard Identification

- Attempted similar angles approach to find corners
- In practice, the outer limits were not the corners
- This approach did not work
Checkerboard Identification

- Attempted similar angles approach to find corners
- In practice, the outer limits were not the corners
- This approach did not work
Checkerboard Identification (cont.)

- Hough Line Transform with a voting scheme worked
- Hough lines with corner detection was optimal
- Fairly robust solution, works well with outliers

Vanishing point voting can fail if lines are parallel
Without vanishing point voting, lines may not lie on checkerboard plane
Camera Calibration

- Camera is calibrated on start-up or file is loaded (3)
- Fortunately, there is a checkerboard present


Camera Calibration (cont.)

- After calibration the video feed is undistorted
Filtering Noise

- Using a running average can mitigate noise
  - Ambient noise can cause false positives
  - A natural solution to final piece placement
Filtering Noise (cont.)

- Piece movement example
Hough Lines

- Edge detection begins with a Canny Edge Detector
- After Canny a Hough Line Transform is applied

```cpp
cv::Canny and cv::HoughLines
```
Hough Line Filtering

- Vanishing point voting is used to group planar lines
- Minima and maxima rho give us interest lines
- Interest line intersections yield good tangent points

Without vanishing point voting, only rho can be used to filter lines, and this can leave lines that do not lie on the checkerboard plane
Initial Corner Detection

- Pose determination requires model matching
- Corners are detected using Shi and Tomasi

```cpp
cv::goodFeaturesToTrack and cv::cornerSubPix```

Board Corner Detection

- Checkerboard corner detection
- Interest line intersection’s closest points are used
Affine Transformation

- Board corners are matched to a model
- An affine transformation is derived and applied

```cpp
cv::getPerspectiveTransform and cv::warpPerspective```

Final Corner Detection

- Homographic image is converted to grayscale
- The corners of the planar board are detected

The original corners could've been transformed!
Hough Circle Detection

- A Gaussian blur is applied
- After the blur a Hough Circle Transform is applied

```cpp
cv::GaussianBlur and cv::HoughCircles```

![Image of Hough Circle Detection with Gaussian blur and Hough Circle Transform applied]
Extracting Game Pieces

- Match if circles centers are close to square centers
- Keep a running average of matches
Beginning the Game

- If user pieces match a template, then game begins
- When game starts add virtual pieces
Innovative Correlation

• Corners are identified with modulus
  o Image is 480 pixels by 480 pixels
  o With 8 squares that is 60 pixels per row and column
  o A good corner has it’s center at modulo 60 +/- a buffer

• Centers are identified with modulus

• Hough line angle voting is done based on modulus
  o Lines are sorted with a custom algorithm
  o Modulus is used to make 0 = 2 * pi = 4 * pi, and pi = 3 * pi = 5 * pi
  o Sorted lines now correlate with direction
Noise Tolerant Piece Detection

- Running average of piece detections (heat map)
- Threshold results to remove anomalies
- An 2-D array of values from 0 to 1
Future Work

• Better corner identification
  o There may be better ways to match a checkerboard
  o It is known that the checkerboard squares have alternating color
  o Template matching?

• Better edge voting
  o There may be better ways to filter the Hough lines, instead of using vanishing point voting combined with min and max rho
  o Linear regression with line of best fit and compare slopes?

• Better piece identification
  o Examining color channels was inferior to grayscale
  o Perhaps some combination of the color channels could be used for piece identification
Questions?