How I learnt computer vision by playing pool

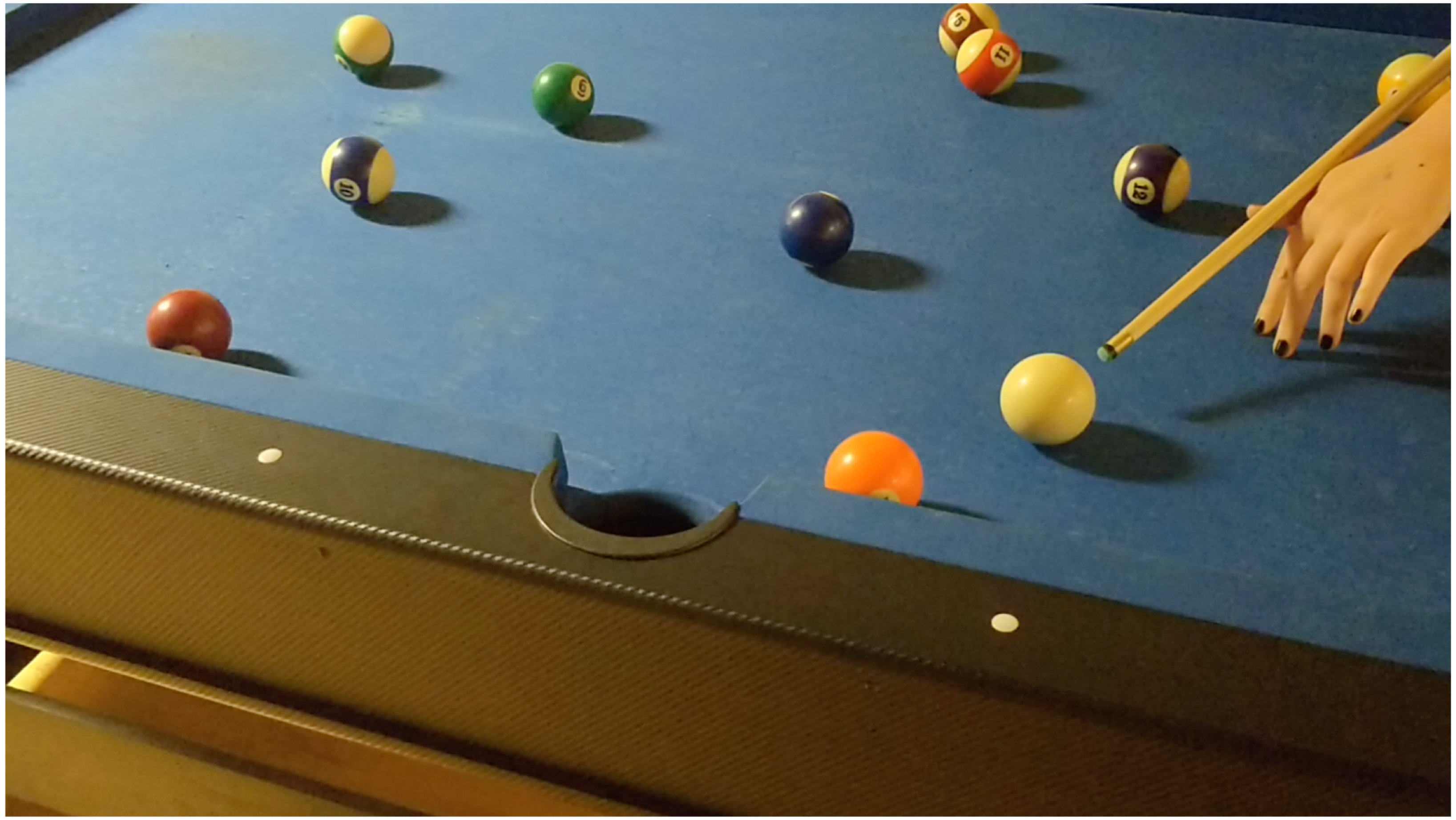
Łukasz Kopeć
Lead Developer (Data Insight), BBOXX

@thektokolwiek

Slides: lukaszkopec.com/files/pydata-pool.pdf

PyData Warsaw, 19 November 2018

Can your pool table do this?



https://youtu.be/SrnoNhOv6h4

Can your pool table do this?

- Real life motivation: an unpredictably uneven pool table
- How to assess fairness in a pool tournament?
- Constraint: no extra setup (e.g. a camera above the table)
  - only use a mobile video of gameplay taken from the side

- Disclaimer: I’m bad at pool
  - you can spin the cue ball *intentionally*‽

How to do it

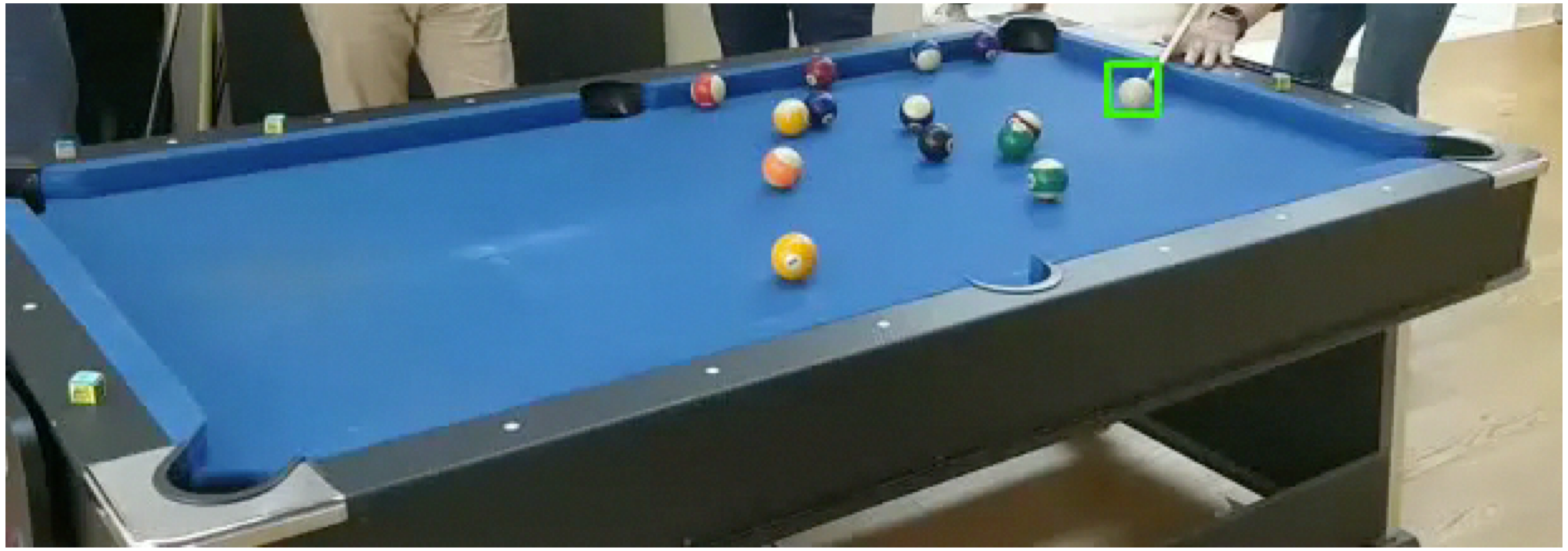
- Detect (and track) balls and table
- Transform into a common reference frame
- Detect collisions (split sequence)
- Quantify how much the table is skewed
- Correlate with results from our pool tournament
- Bonus: Visualise!

How to do it

- **Detect (and track) balls and table**
- Transform into a common reference frame
- Detect collisions (split sequence)
- Quantify how much the table is skewed
- Correlate with results from our pool tournament
- Bonus: Visualise!

OpenCV is pretty good at tracking!

- Caveat: ‘slow’ shots, no occlusions
- Detecting objects is (somewhat) a challenge – works great if you already know where the balls are

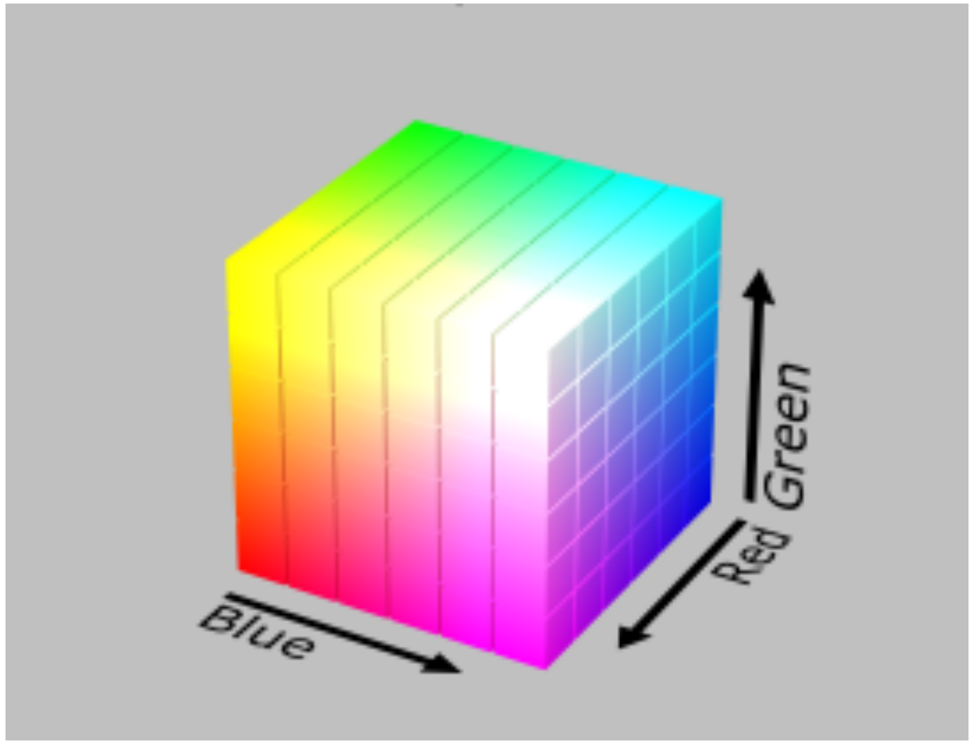


https://youtu.be/rlrxsLQ4ZHc

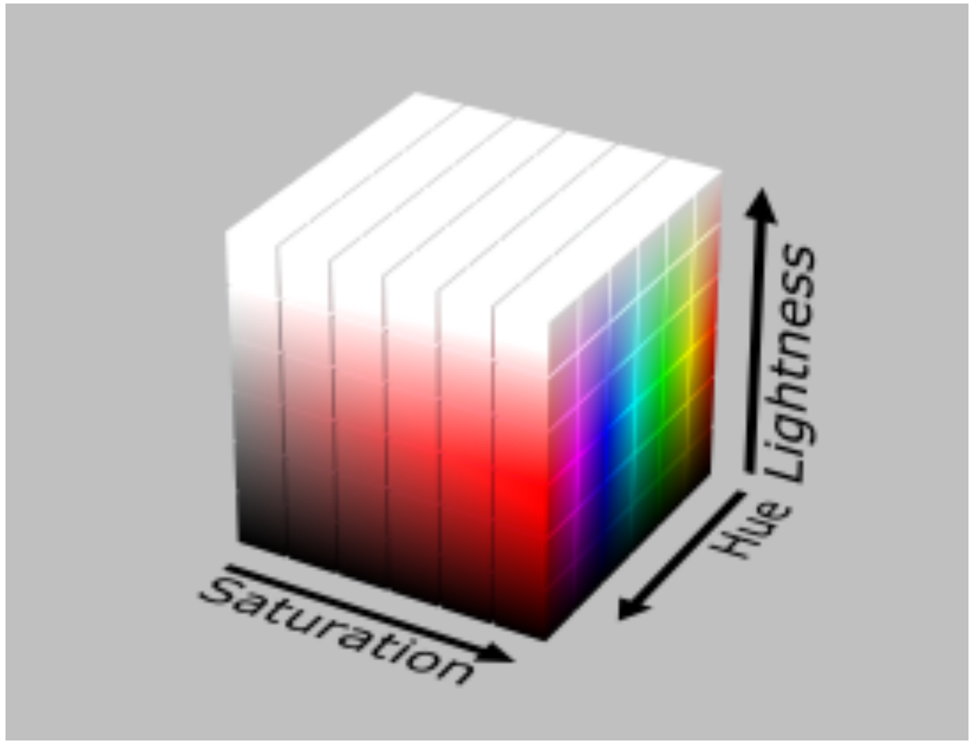
How to do it

- Detect (and track) balls and table
    - Detect ‘blobs’
    - Limit search space to the table
    - Table is ‘that big blue thing in the middle’

RGB < HLS



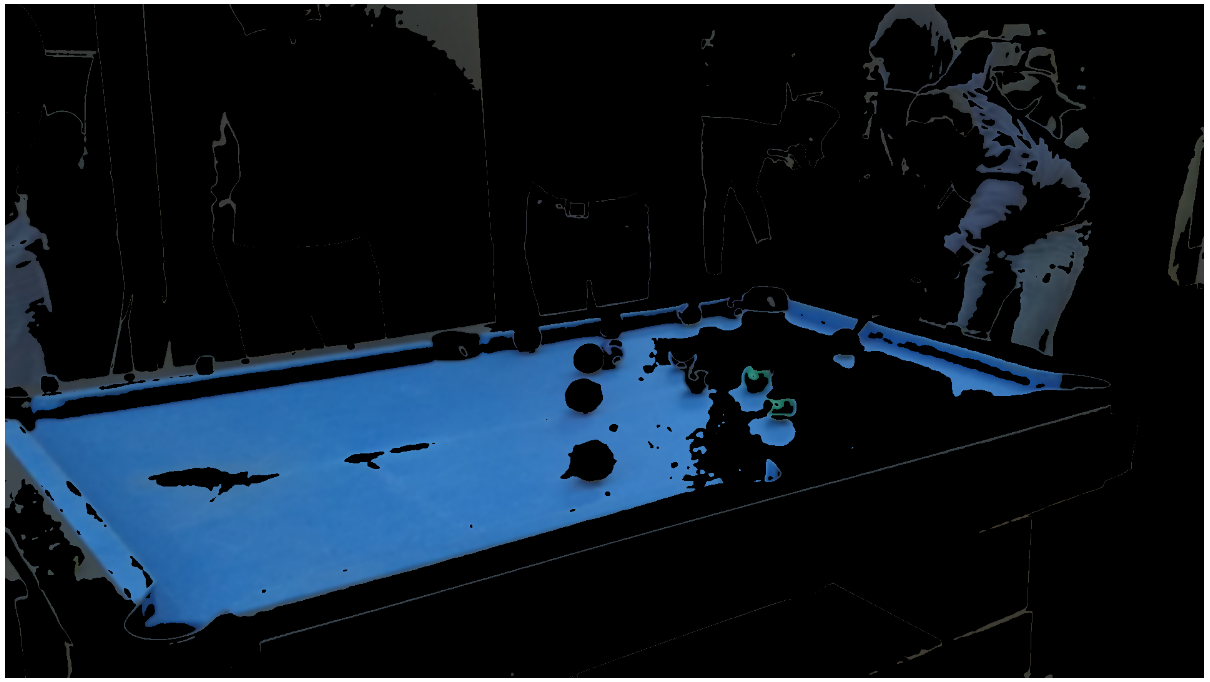
RGB



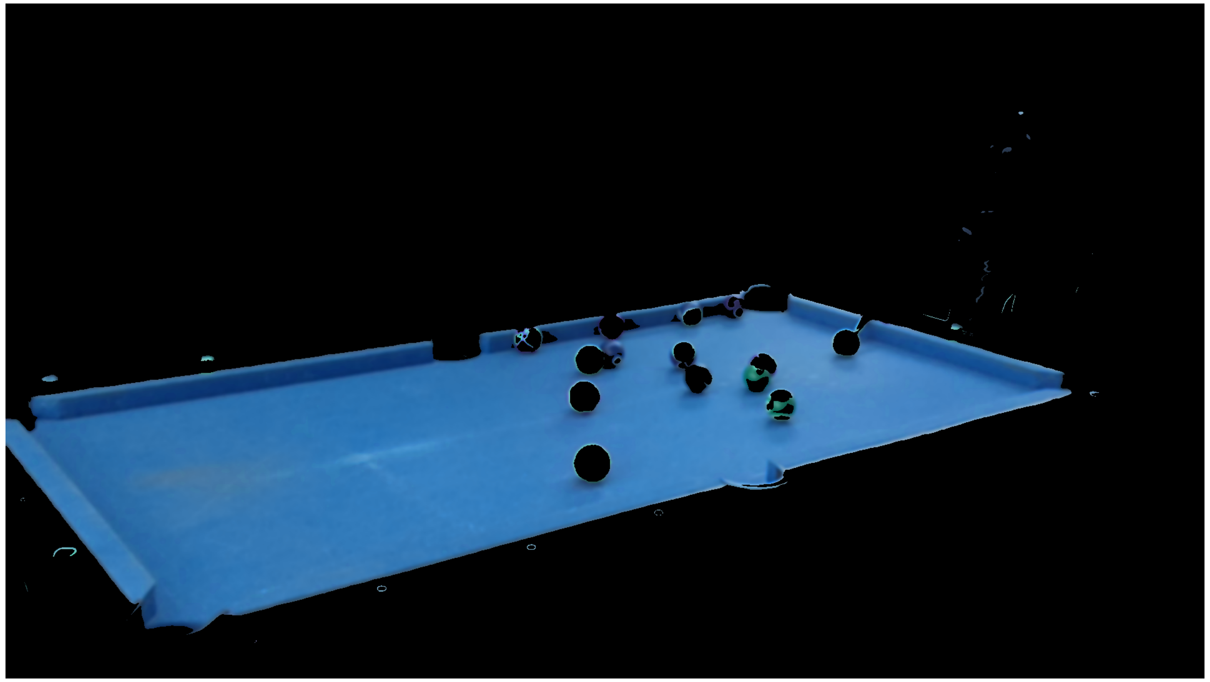
HLS

RGB < HLS

```
cv2.cvtColor(frame, cv2.COLOR_BGR2HLS)
cv2.inRange(frame, min_threshold, max_threshold)
```

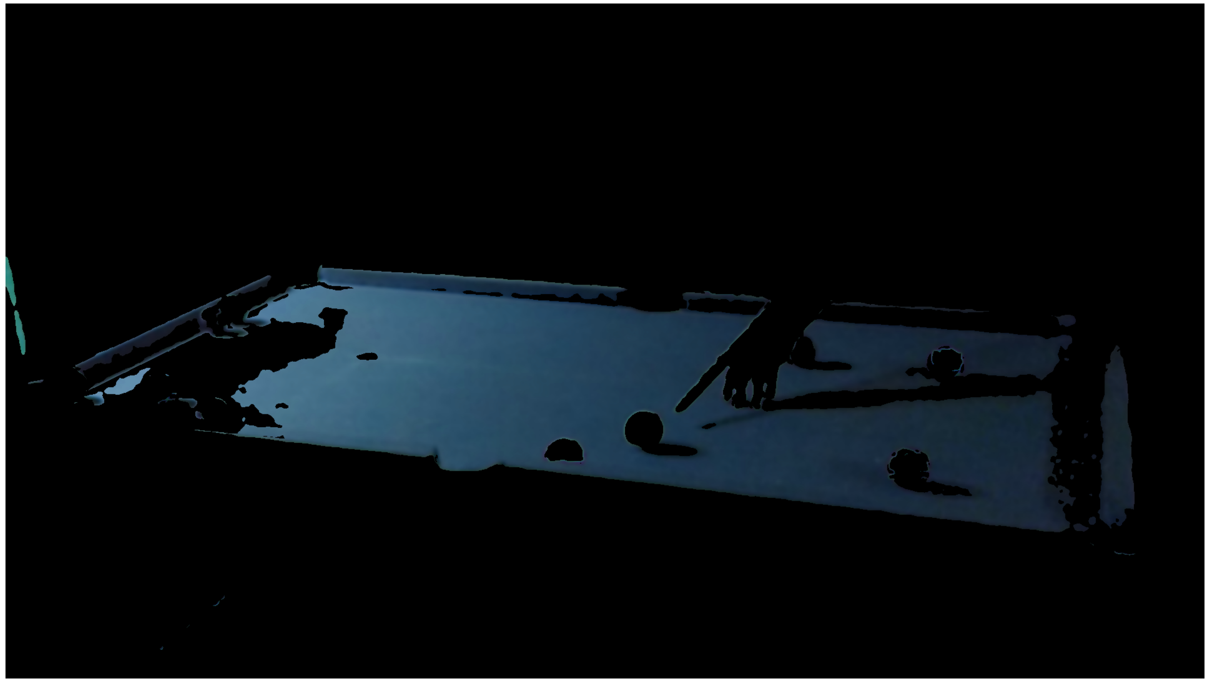


RGB

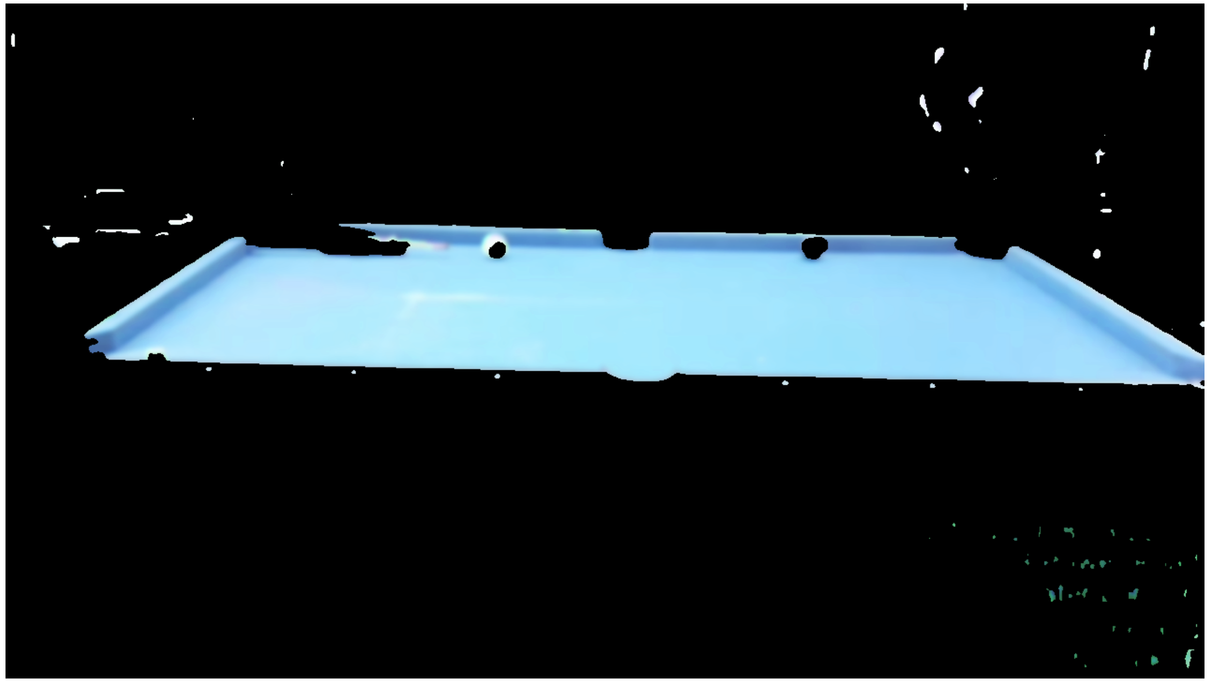


HLS

HLS bonus – works on other videos



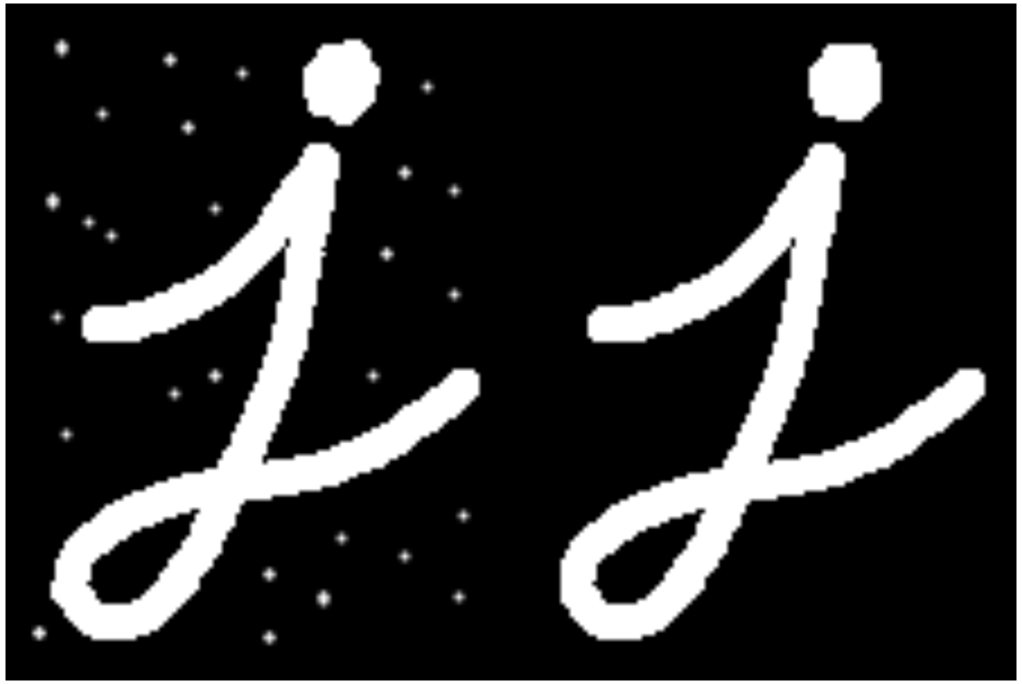
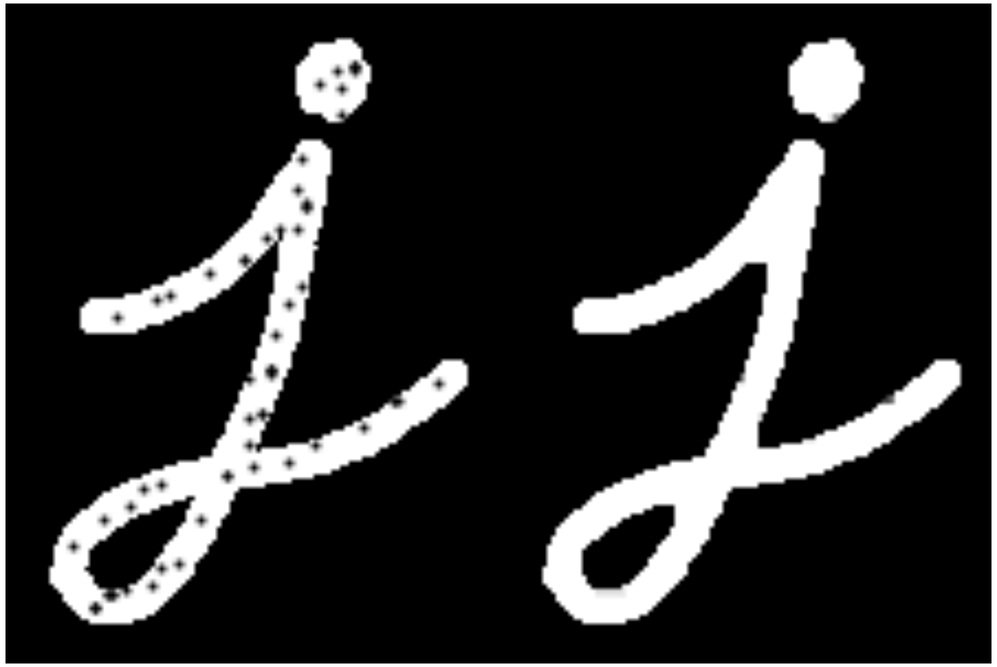
different time of day



different camera

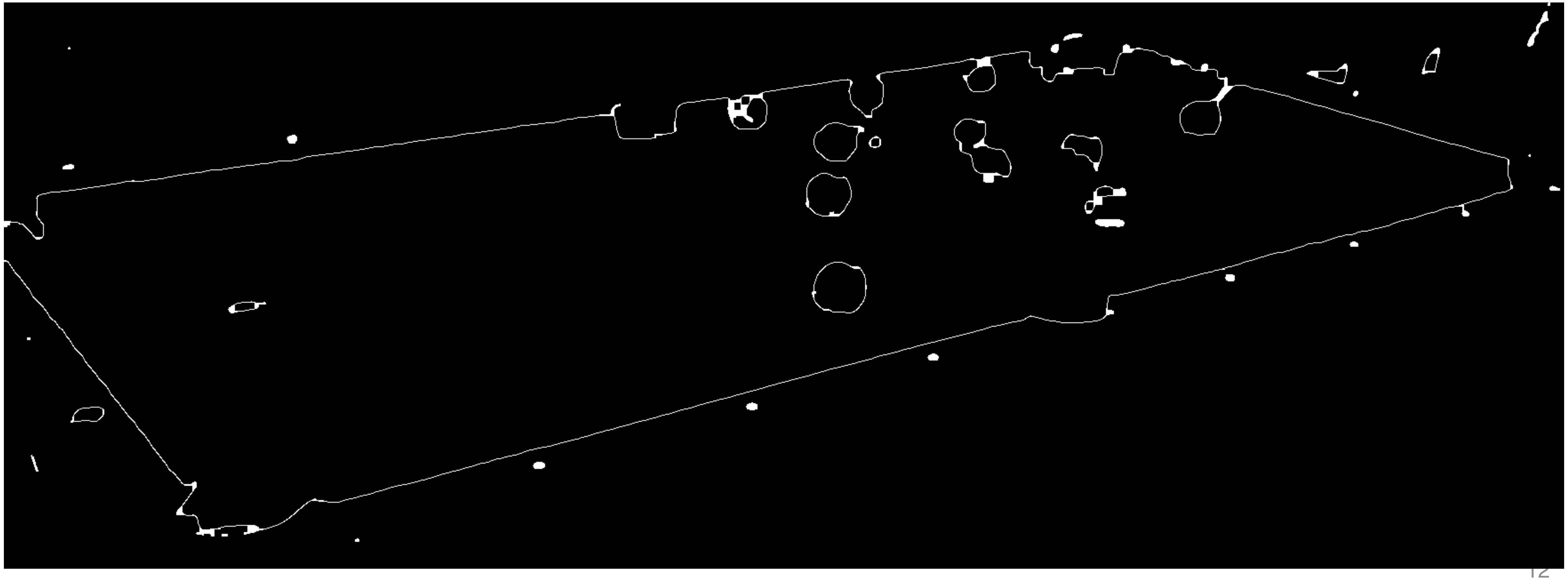
Create one big edge

- Detect (and track) balls and table
  - Canny edge detection -> apply ‘opening’ and ‘closing’ and convex hull

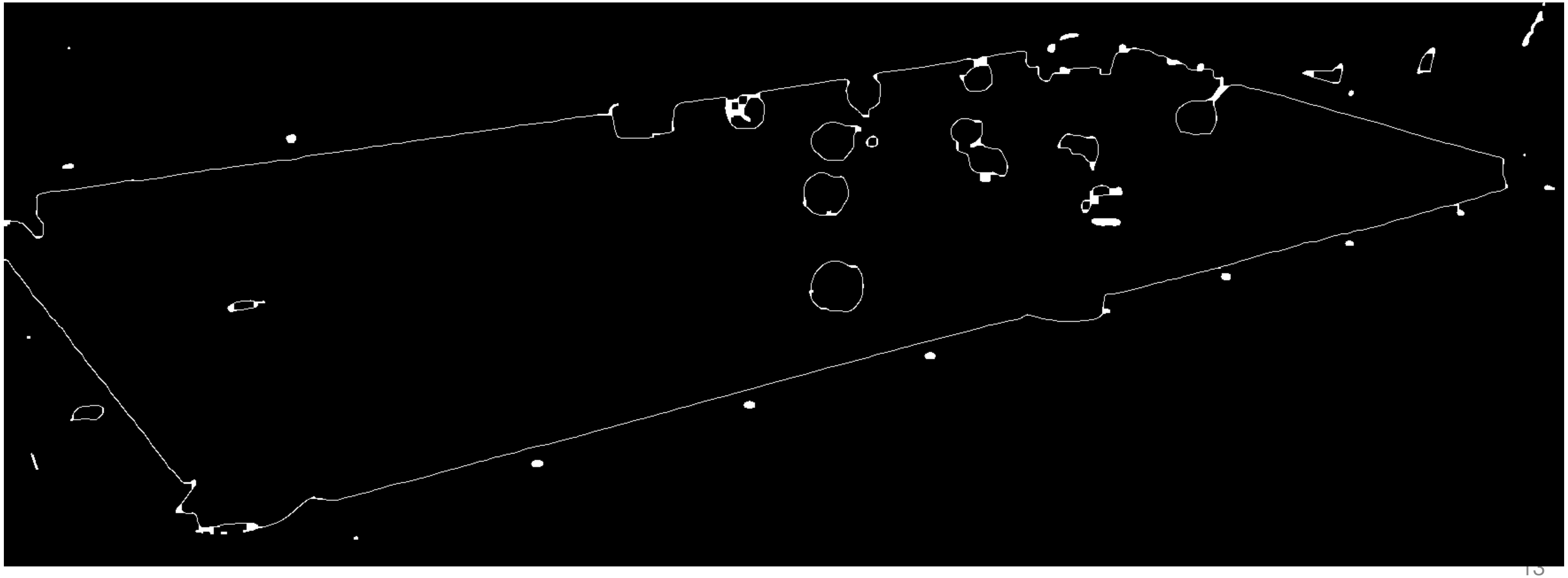
Create one big edge

- Detect (and track) balls and table
    - Canny edge detection -> apply ‘opening’ and ‘closing’ and convex hull



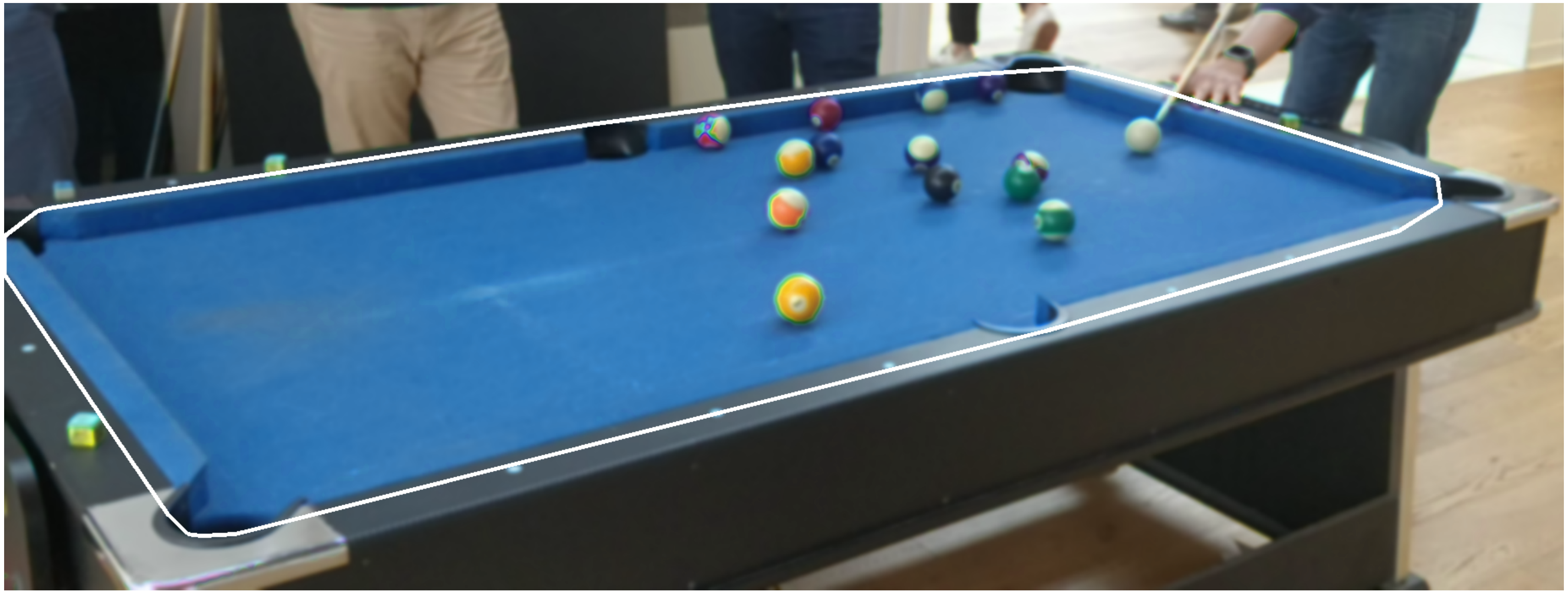
Create one big edge

- Detect (and track) balls and table
  - Canny edge detection -> apply ‘opening’ and ‘closing’ and convex hull



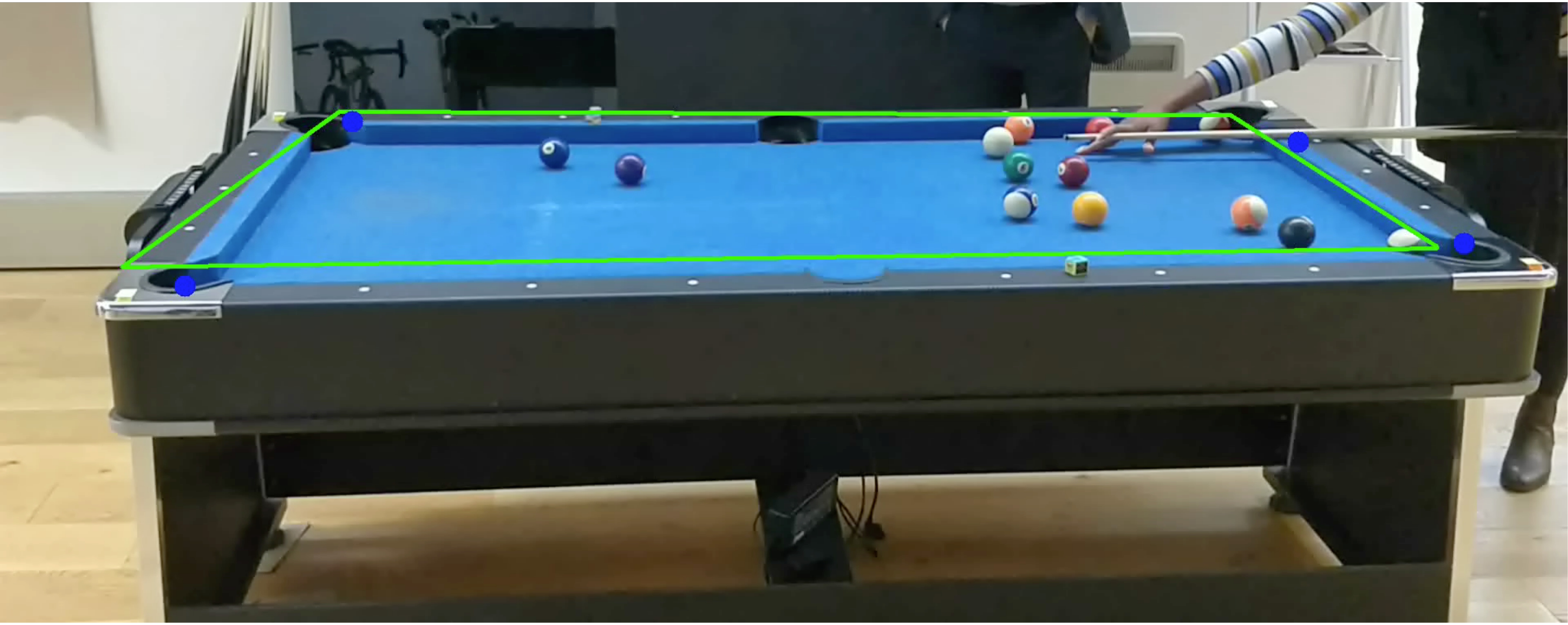
Create one big edge

- Detect (and track) balls and table
  - Canny edge detection -> apply ‘opening’ and ‘closing’ and convex hull



How to do it

- Detect (and track) balls and table
  - SimpleBlobDetector to get the balls’ initial positions
  - Multiple object tracker from OpenCV



https://youtu.be/xJ53Uovtf28

How to do it

- Detect (and track) balls and table
- **Transform into a common reference frame**
- Detect collisions (split sequence)
- Quantify how much the table is skewed
- Correlate with results from our pool tournament
- Bonus: Visualise!

How to do it

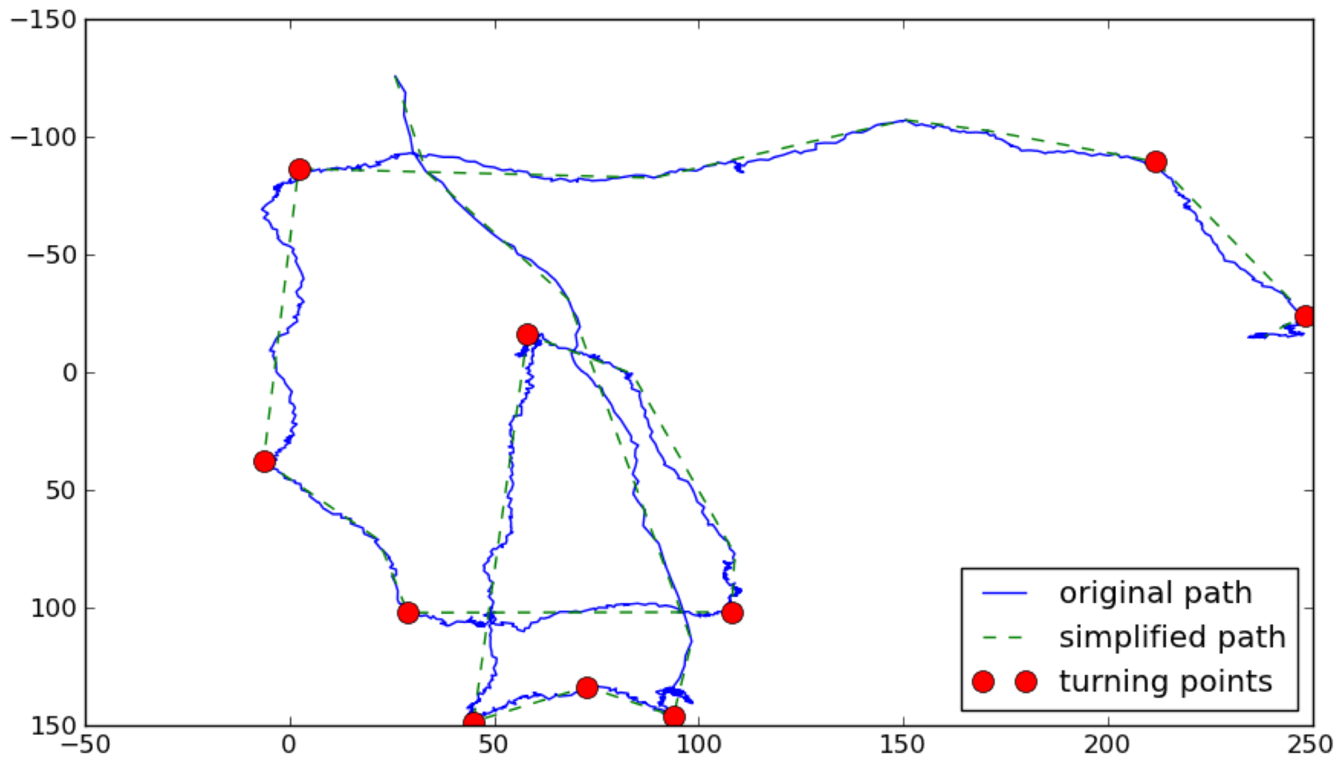
- Transform into a common reference frame
    - Cluster the lines' equations ($y=mx+k$) into four groups
    - Find four corners (line intersections)
    - Transform into a 2:1 rectangle
    - Smooth between frames to avoid jitter

How to do it

- Detect (and track) balls and table
- Transform into a common reference frame
- **Detect collisions (split sequence)**
- Quantify how much the table is skewed
- Correlate with results from our pool tournament
- Bonus: Visualise!

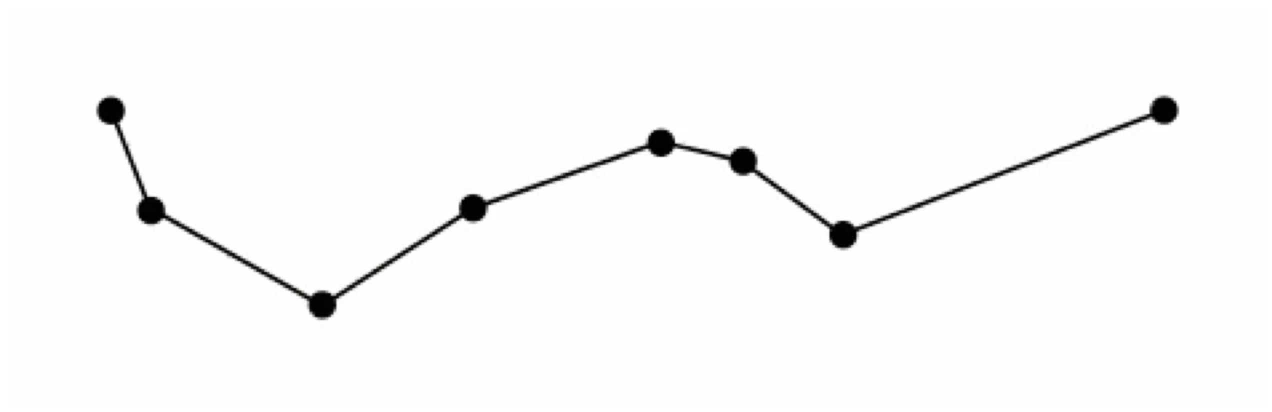
How to do it

- Detect collisions (split sequence)
    - Ramer-Douglas-Peucker (RDP) algorithm to find ‘sharp’ turns



How to do it

- Detect collisions (split sequence)
    - Ramer-Douglas-Peucker (RDP) algorithm to find ‘sharp’ turns



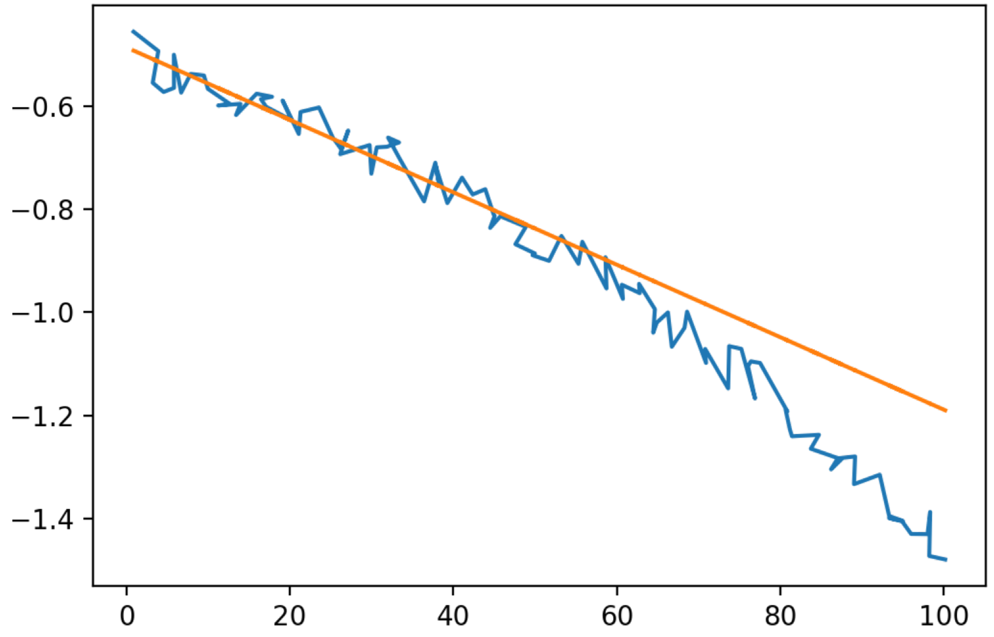
https://en.wikipedia.org/wiki/Ramer%E2%80%93Douglas%E2%80%93Peucker_algorithm#/media/File:Douglas-Peucker_animated.gif

How to do it

- Detect (and track) balls and table
- Transform into a common reference frame
- Detect collisions (split sequence)
- **Quantify how much the table is skewed**
- Correlate with results from our pool tournament
- Bonus: Visualise!

How to do it

- Quantify how much the table is skewed
  - We only care about the last segment (when the ball is moving the slowest)
  - MSE deviation from extrapolated straight line



How to do it

- Detect (and track) balls and table
- Transform into a common reference frame
- Detect collisions (split sequence)
- Quantify how much the table is skewed
- **Correlate with results from our pool tournament**
- Bonus: Visualise!

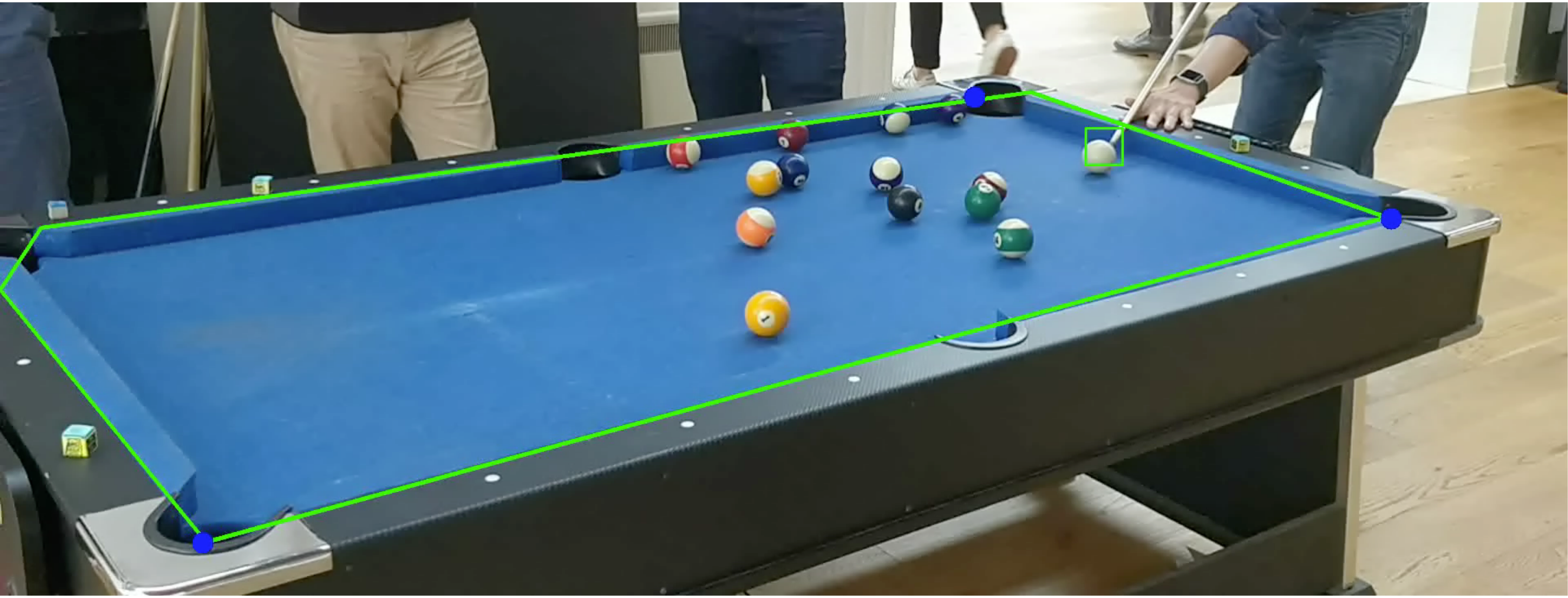
How to do it

- Correlate with results from our pool tournament
  - Hardest part! How do you get people to participate voluntarily?
  - Really low participation rate
  - Game winners ~10% higher skew than game losers, weak correlation with tournament position – does it mean more skewed table is easier?
  - Skill is still a better predictor – if you spin the ball it doesn't go straight!

How to do it

- Detect (and track) balls and table
- Transform into a common reference frame
- Detect collisions (split sequence)
- Quantify how much the table is skewed
- Correlate with results from our pool tournament
- **Bonus: Visualise!**

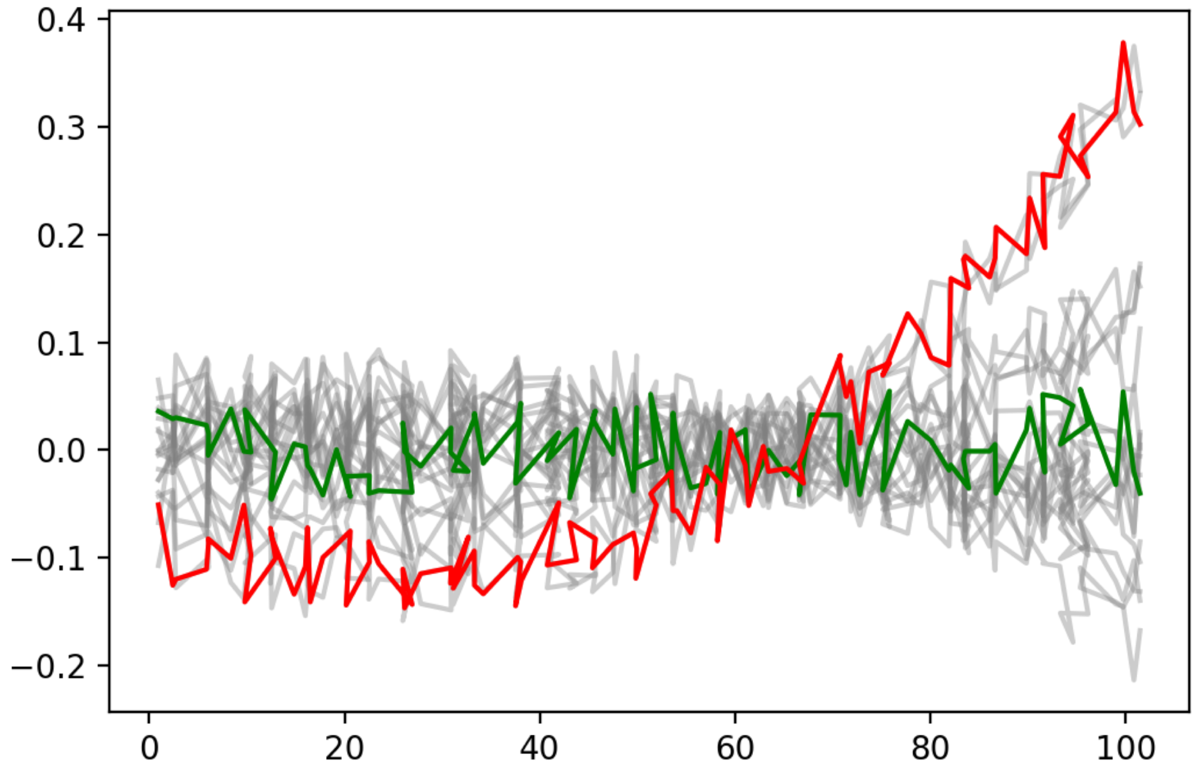
End-to-end



https://youtu.be/NqPKeYX-adI

Average shot

- Rotate the ’predicted direction’ line, scale to unit length



Next steps

- Linear assumption – e.g. norm on a sample of games on a known even table
- Blob detection could do better – how to deal with occlusion?
- Get more (tagged?) data samples

Thank you!

@thektokolwiek

Slides: lukaszkopec.com/files/pydata-pool.pdf