Computer Vision & Pattern Recognition

Course Project

Prof. Kai Hormann

Università della Svizzera italiana

Course Project

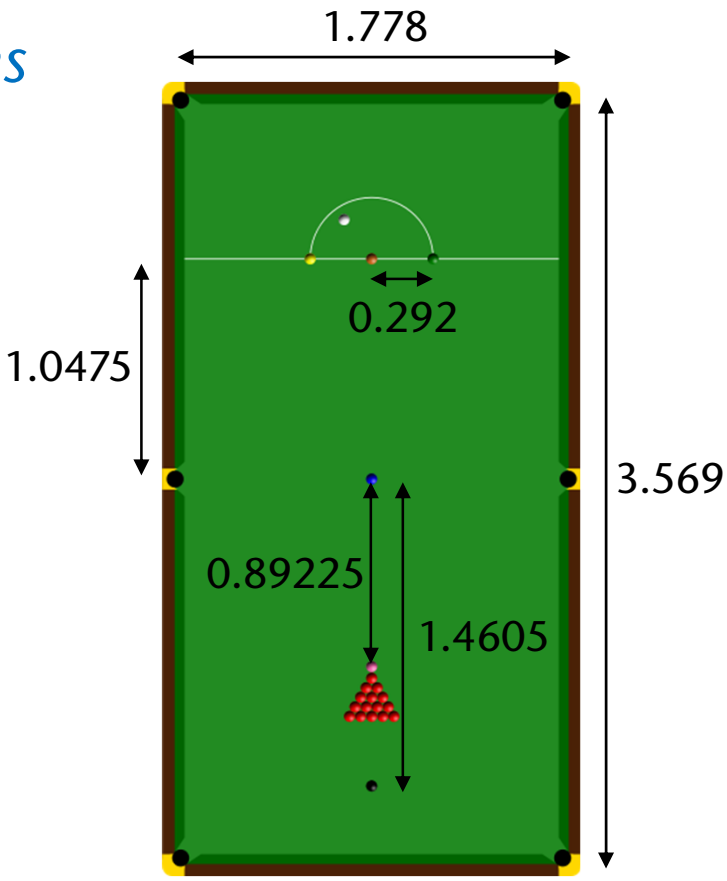
- **Goal**: reconstruct snooker table and balls from side view
- Tasks
  1. pre-process video and filter only those frames that show side view
  2. reconstruct camera position
  3. reconstruct ball positions

Pre-process Video

- take a *reference frame* F_0 that shows the side view
- compute correlation with each other frame F_i
  - e.g., check for each "green" pixel in F_0 if corresponding pixel in F_i is also green and count the matches
  - or, compute some "distance" between F_0 and F_i
  - find a threshold that separates wanted from unwanted frames

Reconstruct Camera

- get *table dimension* and *ball positions*
  - check Wikipedia
  - compute coordinates of the (yellow, green, brown, blue, pink, black) *spots*
  - but what exactly is the *playing area*?
    - the whole “green area”?
    - the area *between* the cushions?
    - measure it in the picture
    - double check in the top-down view
  - compute coordinates of playing area or “green area” corners
    - width and height of the cushions?



Reconstruct Camera

- get corresponding *image points*
  - find frame where all balls are on their spots (e.g. @ 2:11:57)
  - corners of the “green area”
    - use Sobel gradients and Hough transform to find lines between baize (green) and wood (brown) and intersect
  - corners of the playing area
    - harder to find automatically
  - ball marker positions
    - x-coordinates $\rightarrow$ ball centres
    - y-coordinates $\rightarrow$ learn from yellow, green, brown ball and *baulk line* how to relate “top” of the ball to the marker
- note that the image is *symmetric* in x

Reconstruct Camera

- use correspondence pairs to find camera matrix P
  - DLT algorithm minimizes the *algebraic error* $\|A\boldsymbol{p}\|$ subject to $\|\boldsymbol{p}\| = \mathbf{1}$
  - the resulting P can be decomposed to get K, R, $\boldsymbol{C}$
  - alternatively, we can try to minimize the *geometric error*

$$\sum_i \|x_i - PX_i\|^2$$

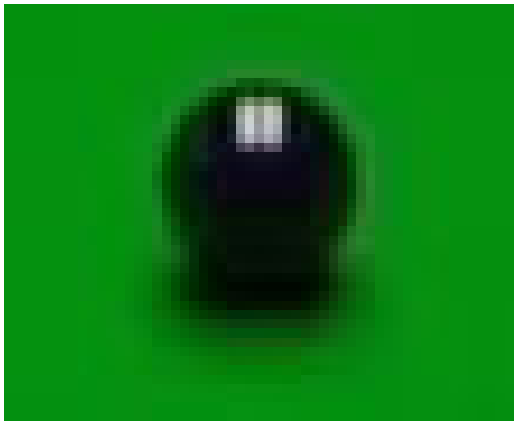
    - especially useful, if we want to work with a constrained K
    - non-linear optimization, requires suitable numerical solvers
- manual approach
  - identify key parameters of $K, R, \boldsymbol{C}$; build P; map X_i into image
  - modify parameters interactively, until you get a good visual match

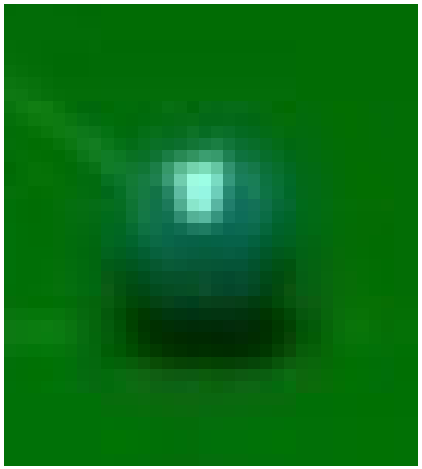
Reconstruct Camera

- additional issues to consider
  - the result can be improved by using the *normalized DLT* (see Sec. 4.4.4 and Sec. 7.1)
  - make sure that the image points of the ball markers on the central vertical line (brown, blue, pink, black) have the same *cross ratio* as the corresponding world points

Reconstruct Ball Positions

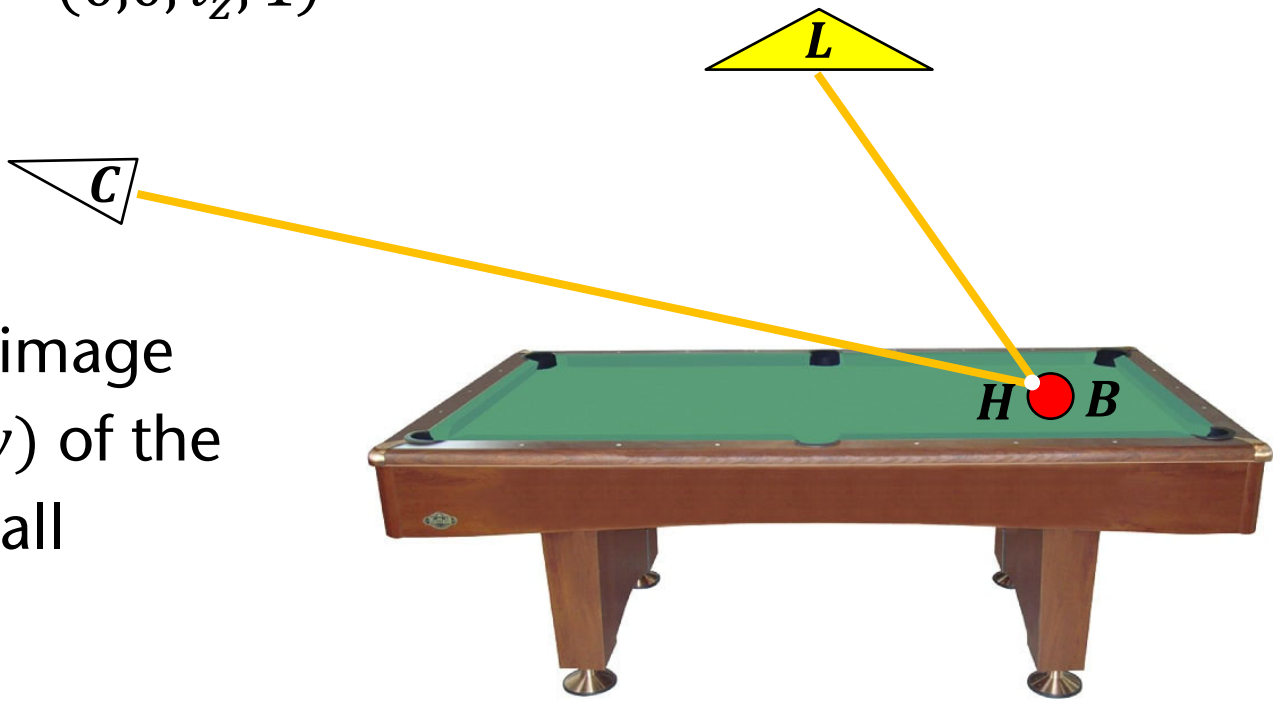
- did you notice the white reflection on the balls?
- caused by the overhead lights
  - two long white lights
  - length: approximately 4 meters
  - height: approximately 5–6 meters (check video @ 2:11:22)
- can be used to estimate the ball position





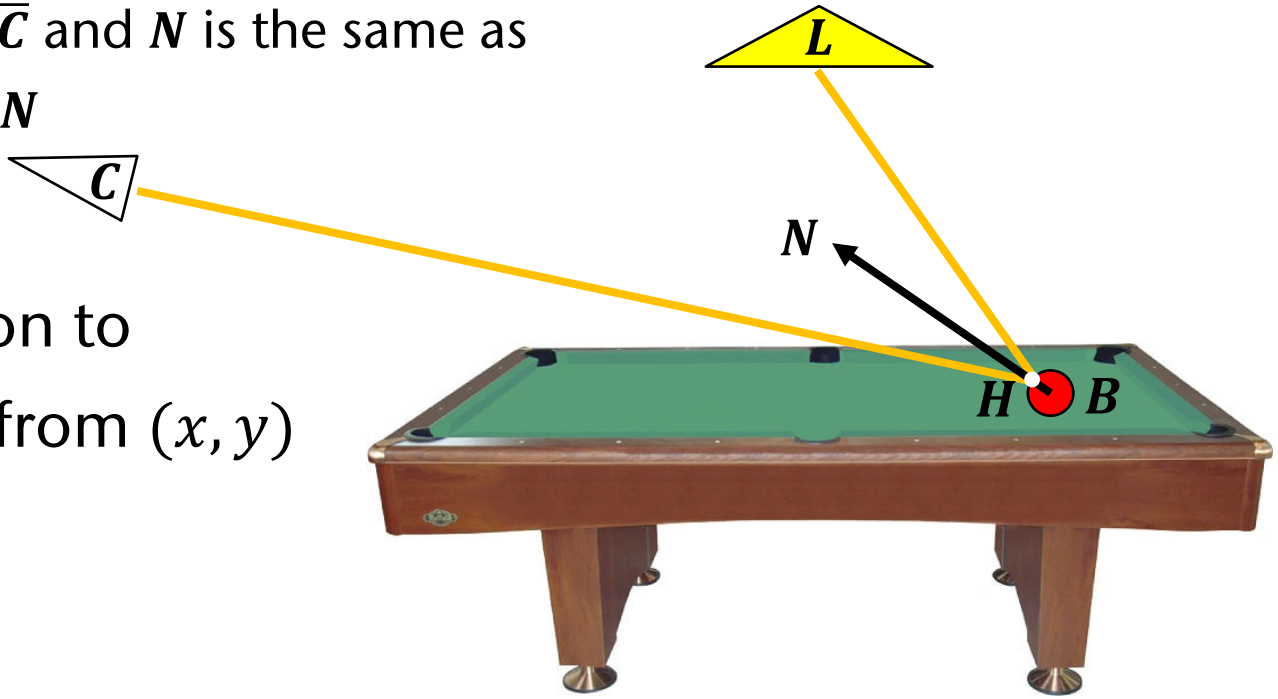
Reconstruct Ball Positions

- ball centre position $\boldsymbol{B} = \left(b_x, b_y, 0.02625, 1\right)^T$
- camera position $\boldsymbol{C} = \left(0, c_y, c_z, 1\right)^T$
- light position $\boldsymbol{L} = (0, 0, l_z, 1)^T$
- highlight $\boldsymbol{H}$
- c_y, c_z, l_z known
- **Goal**: given the image coordinates (x, y) of the highlight, find ball position (b_x, b_y)



Reconstruct Ball Positions

- *first step:* given (b_x, b_y), work out (x, y)
- obey the Geometry
  - $\boldsymbol{C}, \boldsymbol{B}, \boldsymbol{L}, \boldsymbol{H}$, and the ball normal $\boldsymbol{N}$ lie in the same plane
  - angle between $\overline{\boldsymbol{HC}}$ and $\boldsymbol{N}$ is the same as between $\overline{\boldsymbol{HL}}$ and $\boldsymbol{N}$
- *second step:*
  invert this relation to
  recover (b_x, b_y) from (x, y)



Reconstruct Ball Positions

- remains to detect the (centres of the) highlights in the image
  - find red balls
  - find (approximately contiguous) white region within red region
  - compute average position of these white pixels
- alternatively
  - use *correlation* with prototypical image of red ball
  - *adapt size*, based on image y-coordinate
  - **problem**: partial occlusion
  - **solution**: match only upper 1/3 of ball image