Fast Camera Calibration for the Analysis of Sport Sequences

Dirk Farin,
Jungong Han
Peter H. N. de With

Contact address:
Dirk Farin
Eindhoven University of Technology (TU/e)
Design Technology Institute
5600 MB, Eindhoven, The Netherlands
d.s.farin@tue.nl

Introduction

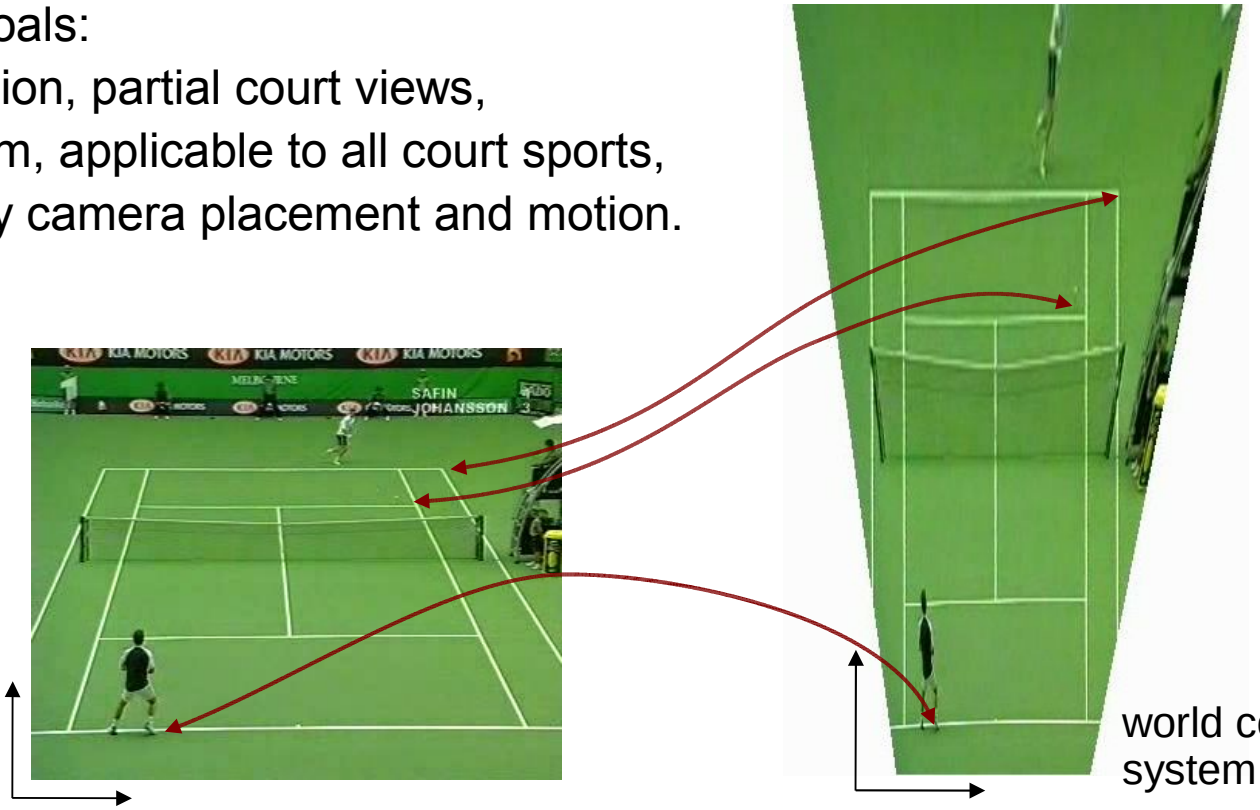
Semantic analysis of sport videos depends on

- information about player and ball position,
- position in real-world coordinates, not screen coordinates.

Geometric mapping between screen coordinates and real-world required.

Algorithm design goals:

- robust to occlusion, partial court views,
- generic algorithm, applicable to all court sports,
- support arbitrary camera placement and motion.



Camera Model

- 3D camera set-up and image formation (in homogeneous coordinates):

$$\mathbf{p_i} = \mathbf{H}\mathbf{p_i'} = \underbrace{\begin{pmatrix} f & 0 & o_x \\ 0 & f & o_y \\ 0 & 0 & 1 \end{pmatrix}}_{\text{camera projection}} \underbrace{\begin{pmatrix} r_{00} & r_{01} & r_{02} & t_x \\ r_{10} & r_{11} & r_{12} & t_y \\ r_{20} & r_{21} & r_{22} & t_z \end{pmatrix}}_{\text{camera rotation and placement}} \underbrace{\begin{pmatrix} x' \\ y' \\ z' = 0 \\ 1 \end{pmatrix}}_{\text{world coordinate}}$$

assume that court ground is at z'=0

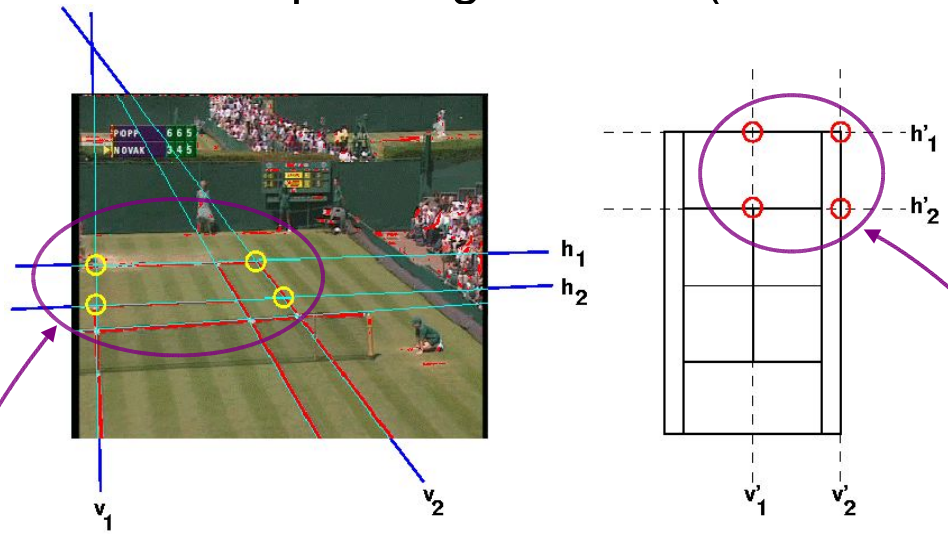
- multiplying this out leads to 8-parameter perspective motion model:

$$x' = \frac{h_{00}x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + h_{22}}, \quad y' = \frac{h_{10}x + h_{11}y + h_{12}}{h_{20}x + h_{21}y + h_{22}}$$

- 8 parameters ➙ 8 equations required to solve for parameters
- ➙ 4 point correspondences required (each gives 2 equations)

Principle of Calibration Algorithm

- Assume that four corresponding features (calibration points) are known.



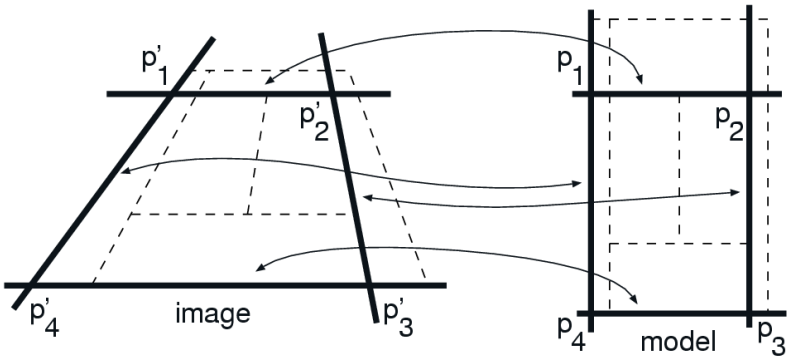
- Use point correspondences to solve for camera parameters:

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & x_1x'_1 & y_1x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & x_1y'_1 & y_1y'_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & x_2x'_2 & y_2x'_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & x_2y'_2 & y_2y'_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & x_3x'_3 & y_3x'_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & x_3y'_3 & y_3y'_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & x_4x'_4 & y_4x'_4 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & x_4y'_4 & y_4y'_4 \end{pmatrix} \boxed{\begin{pmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \end{pmatrix}} = \begin{pmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ x'_3 \\ y'_3 \\ x'_4 \\ y'_4 \end{pmatrix}$$

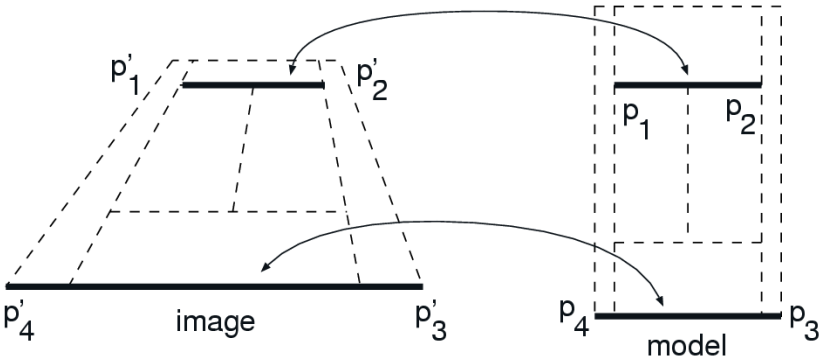
- Calibration points should be selected depending on the image content.

Calibration Point Detection

- Detection of calibration **points** is unreliable
  - occlusion by players
  - outside of visible image
  - difficult detection
- Detection of lines is more reliable.
  - Method 1 (robust): compute 4 intersection points from 4 lines.
  - Method 2 (fast): take the end-points of 2 line segments.



method 1 (robust)

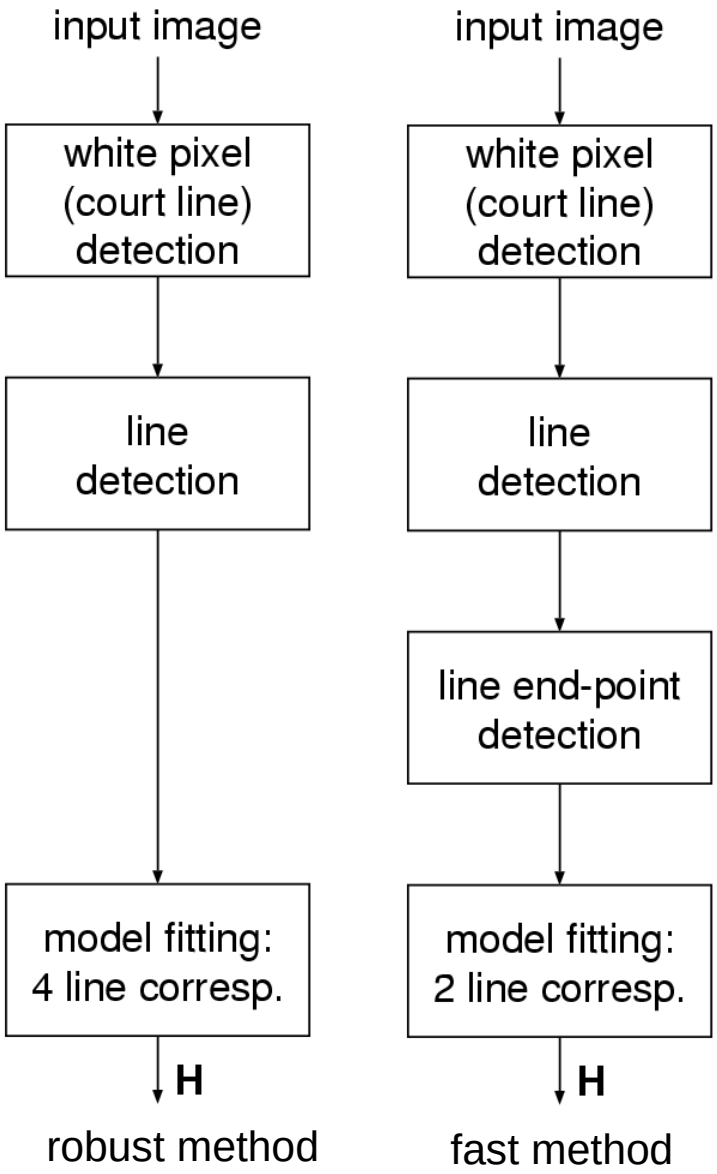


method 2 (fast)

Flowgraph of Camera Calibration Algorithm

- Both methods share same preprocessing:
  - Detect white pixels (belonging to court lines).
  - Detect court lines (in parametric form).

- Model fitting step:
  - Iterate through lines in court model.
  - Iterate through lines in image.
  - For each set of corresponding lines:
    - hypothesize transformation **H**
    - evaluate fitting quality
  - Choose **H** with best fitting quality.

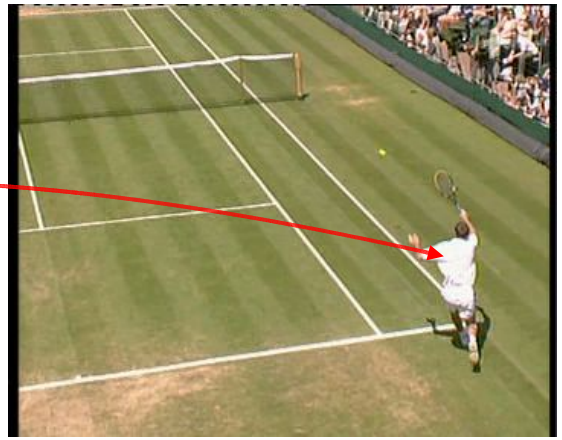


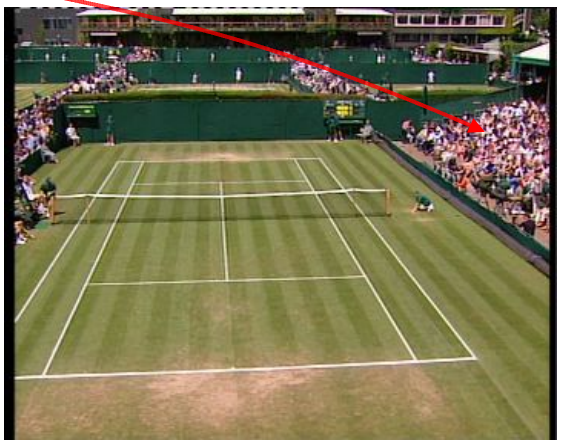
White (Court-Line) Pixel Detection (1/3)

Detect white pixels belonging to court-lines.

Challenges:

- Players often dressed in white
  (large white regions)
- White colors in the audience, stadium
  (fine textured areas)





Each pixel must pass several tests:

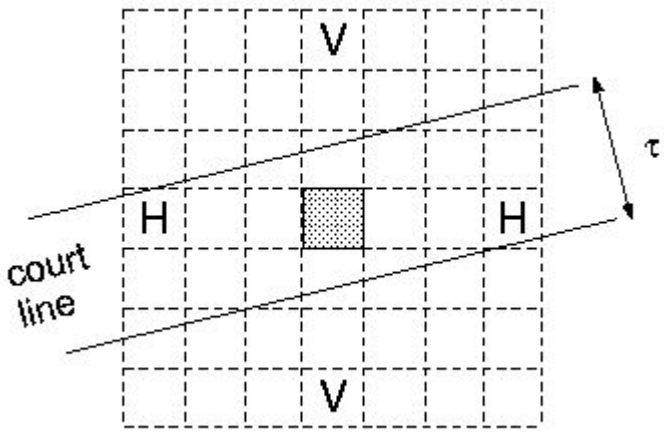
- above fixed brightness threshold ?
  - exclude dark regions
- brighter than neighborhood ?
  - exclude homogeneous areas
- linear structure in neighborhood ?
  - exclude textured areas

Tests have increasing complexity.

White (Court-Line) Pixel Detection (2/3)

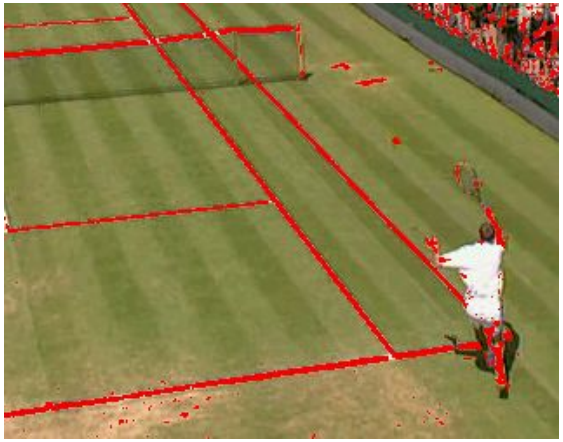
Flat area exclusion rule (test 2):

- pixel candidate must be brighter than
  both pixels marked 'H', or
  both pixels marked 'V'.



- Exclusion of homogeneous white areas
  with very low computational cost.

input frame



remaining pixel candidates

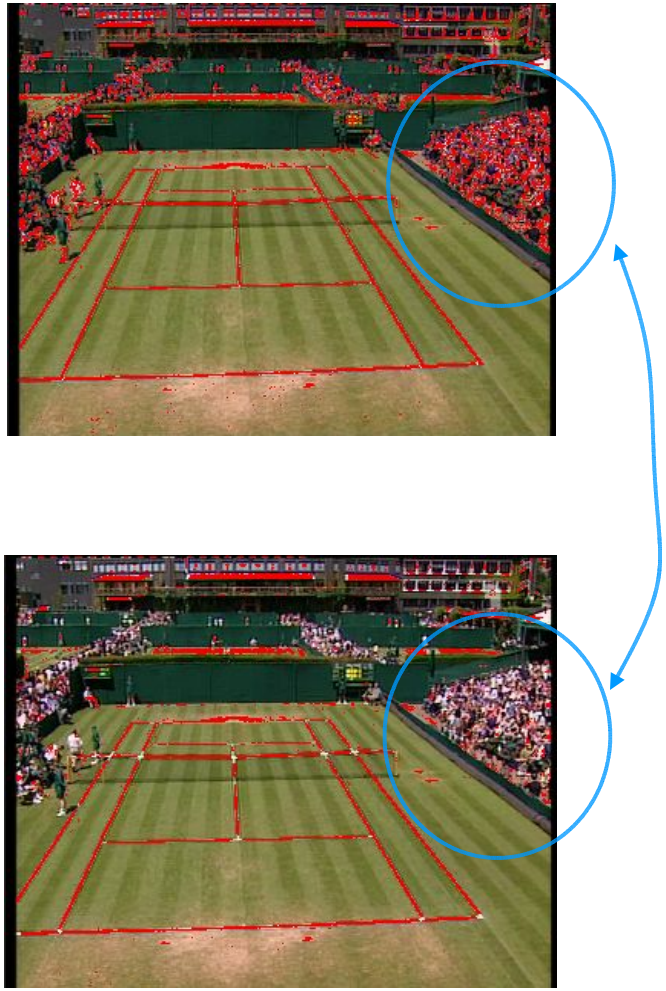
White (Court-Line) Pixel Detection (3/3)

Textured area exclusion rule (test 3):

- compute structure matrix within the pixel neighborhood:

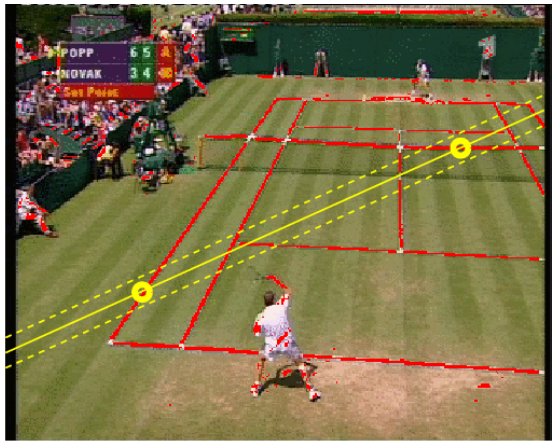
$$\mathbf{S} = \begin{pmatrix} J_{xx} & J_{xy} \\ J_{xy} & J_{yy} \end{pmatrix} = \sum_{x=p_x-b}^{p_x+b} \sum_{y=p_y-b}^{p_y+b} \nabla g(x,y) \cdot (\nabla g(x,y))^T$$

- structure can be classified by evaluating the magnitude of the two eigenvalues.
  - $\lambda_1 >> \lambda_2 \rightarrow$ linear structure
- complex test, but has to be applied only to remaining candidate pixels

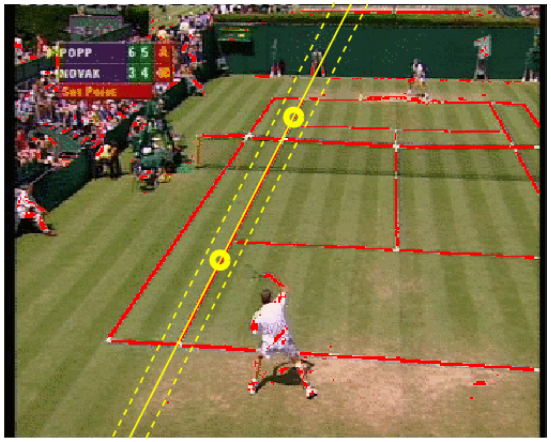


Court-Line Detection (1/2: Continuous Lines)

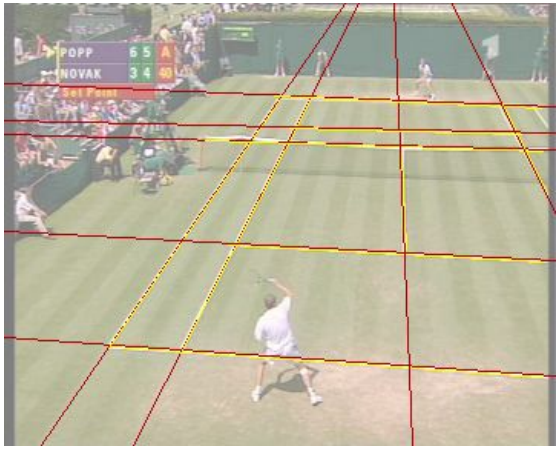
- Use RANSAC to find lines in set of white pixels.
  - Randomly select two white pixels.
  - Hypothesize line through these two pixels.
  - Compute support of the hypothesized line (white pixels near line).
  - Iterate and select line with largest support.
- Remove white pixels near line from input, iterate to find more lines.



sample with little support (no court line)



sample with high support (line detected)

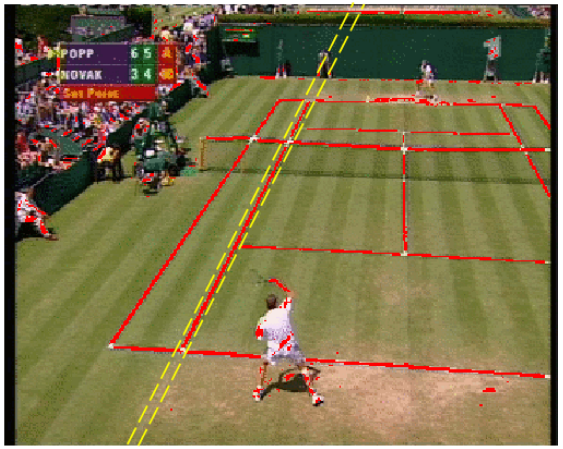


all detected lines (red) supporting pixels (yellow)

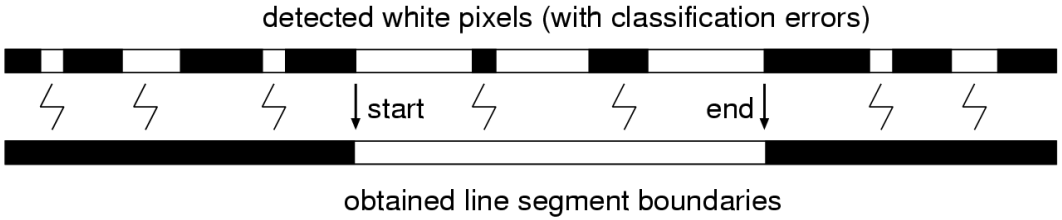
Court-Line Detection (2/2: End-Point Detection)

- Determine end-points to get a line segment.
- White pixels on line-segment can be missing:
  - occlusion by players,
  - detection errors,
  - physical end of line is outside of image.

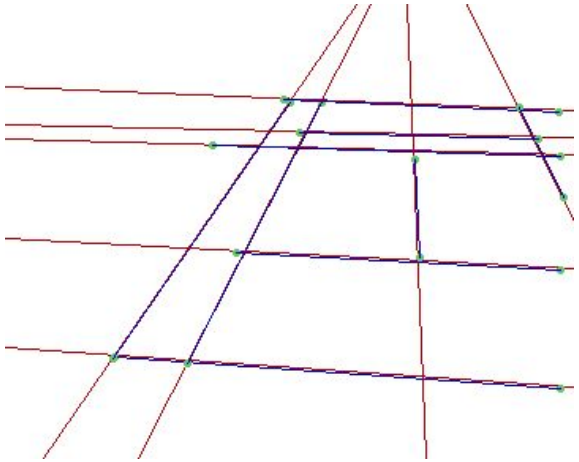
- Ideally, all pixels within the line-segment should be white, and
- all pixels before and after the segment should be non-white (*black*).
- Find *start* and *end* such that number of errors is minimized.



1) scanning for white pixels along the line



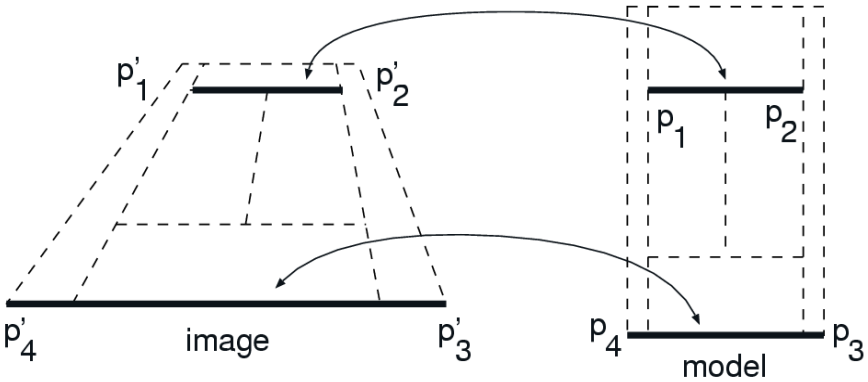
2) determine end-points to minimize number of errors



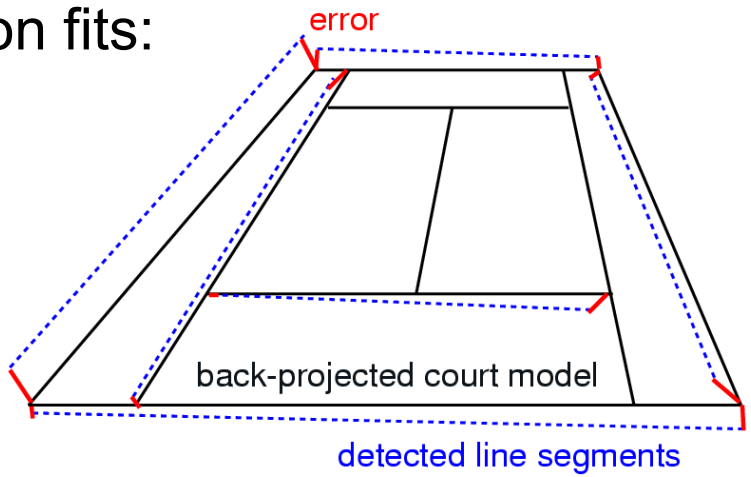
3) detected line segments (blue)

Court Model Fitting (Fast Method)

- Iterate through
  - all line pairs in input image, and
  - all line pairs in the court model.
- On each side, we know the position of four end-points.
- Compute transformation **H**.



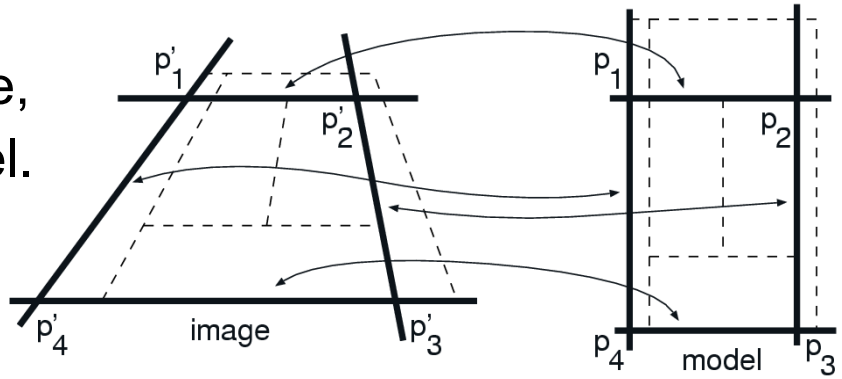
- Check if hypothesized transformation fits:
  - Project court model back to image coordinates.
  - Compute distance of all lines to nearest detected segments.



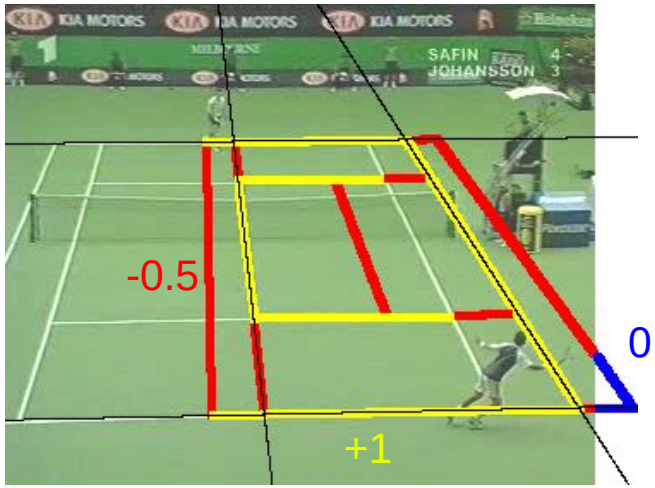
- advantage: fast
- disadvantage: relies on accurate detection of line ends, fails if only part of the court is visible

Court Model Fitting (Robust Method)

- Iterate though
  - all 4-sets of lines in the input image,
  - all 4-sets of lines in the court model.
- Compute four intersection points.
- Compute transformation **H**.



- Check if hypothesized transformation fits:
  - Project court model back to image coordinates.
  - Count white pixels near court lines.

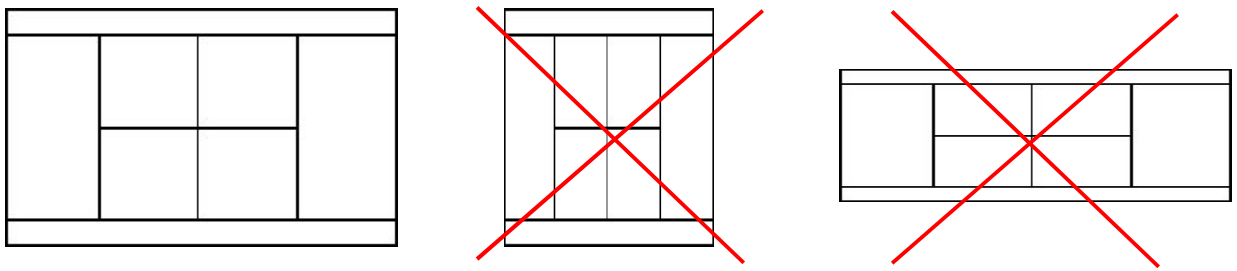


- advantage: works is most cases
- disadvantage: slower, because many configurations are checked

Quick Hypothesis Rejection

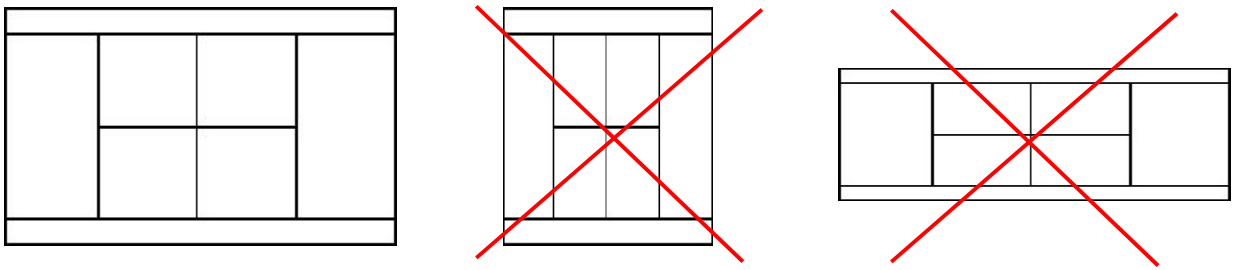
- Computation of fitting cost is computationally expensive.

- Use quick rejection tests to omit testing of most configurations.
- Check:
  - Area: court size should not be too small.
  - Isotropic scaling: aspect ratio of court should not change.



Quick Hypothesis Rejection (Aspect Ratio 1/2)

- Camera model allows non-isotropic scaling, which is impossible in the real world.



- Non-isotropic scaling in our camera model:

$$\mathbf{p_i} = \mathbf{H}\mathbf{p}'_\mathbf{i} = \underbrace{\begin{pmatrix} f & 0 & o_x \\ 0 & f & o_y \\ 0 & 0 & 1 \end{pmatrix}}_{\text{internal camera parameters}} \underbrace{\begin{pmatrix} r_{00} & r_{01} & r_{02} & t_x \\ r_{10} & r_{11} & r_{12} & t_y \\ r_{20} & r_{21} & r_{22} & t_z \end{pmatrix}}_{\text{camera rotation, translation}} \underbrace{\boxed{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \beta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}}_{\text{non-isotropic scaling}} \begin{pmatrix} x' \\ y' \\ z' = 0 \\ 1 \end{pmatrix}$$

β should always be equal to 1

Quick Hypothesis Rejection (Aspect Ratio 2/2)

- Decompose transformation matrix H to obtain β.
- Assume that principal point is known.

$$\begin{pmatrix} 1 & 0 & -o_x \\ 0 & 1 & -o_y \\ 0 & 0 & 1 \end{pmatrix} \mathbf{H} = \mathbf{H}' = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{00} & r_{01} & t_x \\ r_{10} & r_{11} & t_y \\ r_{20} & r_{21} & t_z \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} f r_{00} & \beta f r_{01} & f t_x \\ f r_{10} & \beta f r_{11} & f t_y \\ r_{20} & \beta r_{21} & t_z \end{pmatrix}$$

- We know that { r_i } is orthonormal (rotation matrix).

these are our camera parameters (except a scaling factor)

$$\frac{h'^2_{00}}{f^2} + \frac{h'^2_{10}}{f^2} + h'^2_{20} = \frac{h'^2_{01}}{\beta^2 f^2} + \frac{h'^2_{11}}{\beta^2 f^2} + \frac{h'^2_{21}}{\beta^2}$$

$$\frac{h'_{00} h'_{01}}{\beta f^2} + \frac{h'_{10} h'_{11}}{\beta f^2} + \frac{h'_{20} h'_{21}}{\beta} = 0$$

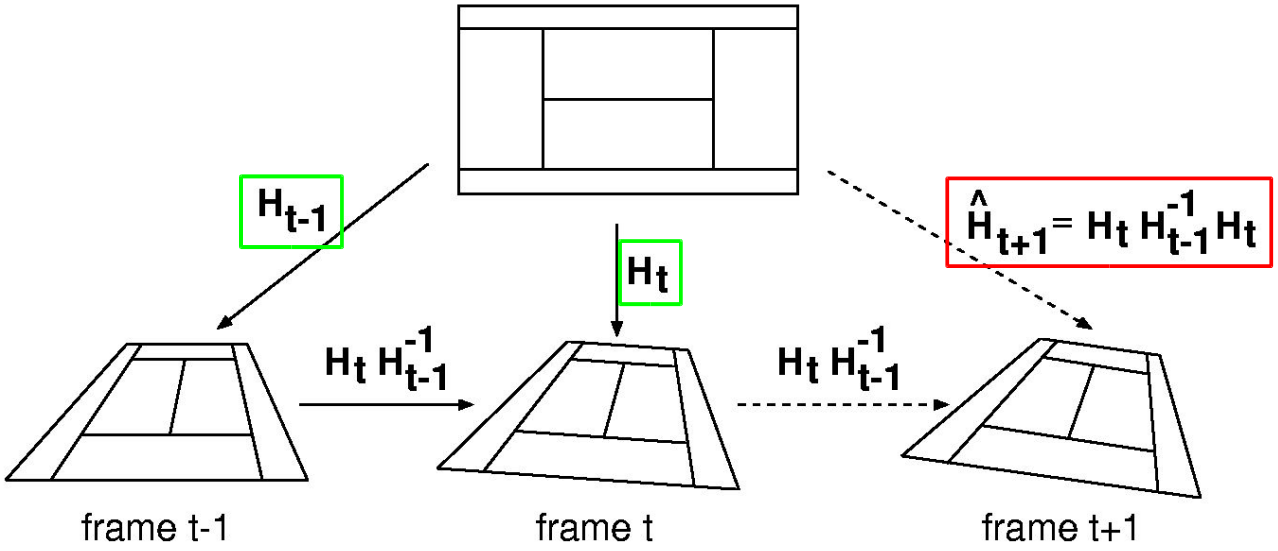
- Hence, we obtain β as:

$$\beta^2 = \frac{h'^2_{01} + h'^2_{11} + f^2 h'^2_{21}}{h'^2_{00} + h'^2_{10} + f^2 h'^2_{20}} \quad ; \quad f^2 = -\frac{h'_{00} h'_{01} + h'_{10} h'_{11}}{h'_{20} h'_{21}}$$

- Consider a parameter set as invalid if β<0.5 or β>2.0

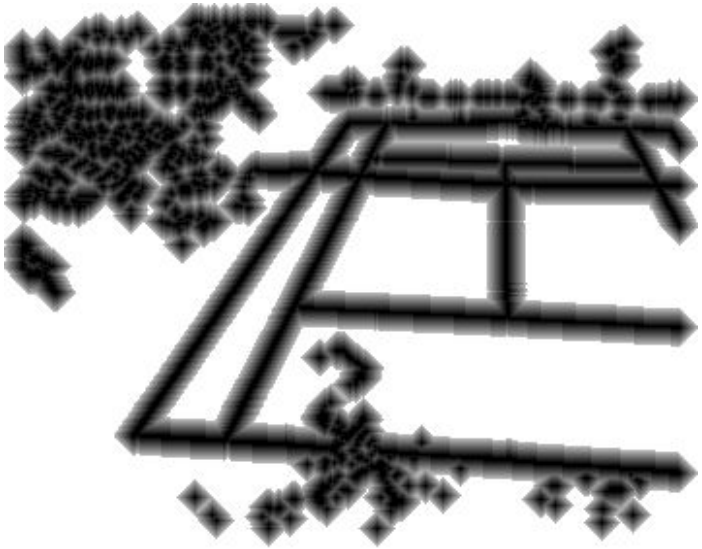
Model Tracking (1/2)

- Algorithm so far operated on single frames.
- When applied on a video-sequence, previous parameter sets can be used to compute an initial estimate.



Model Tracking (2/2)

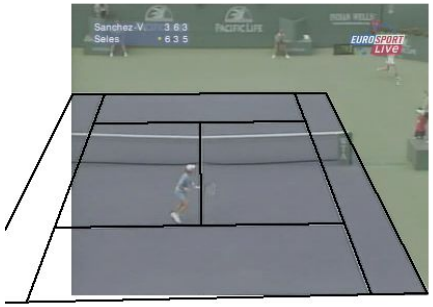
- Optimize camera parameters by minimizing distance between model lines and court-line pixels.
- Solution with gradient-descent algorithm.

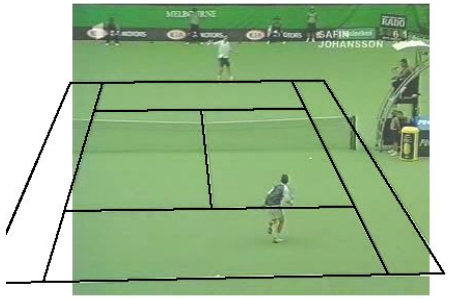


pixel cost (summed along court lines)

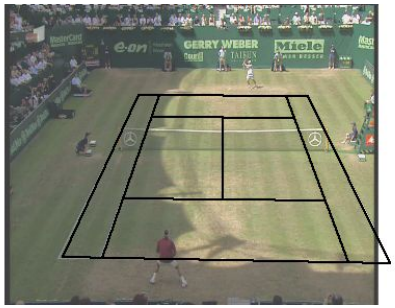
Results (*Tennis*; 1/2)

- different court types:

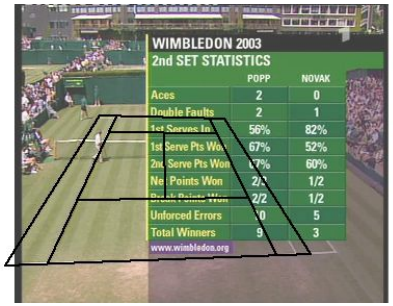




- strong shadow:

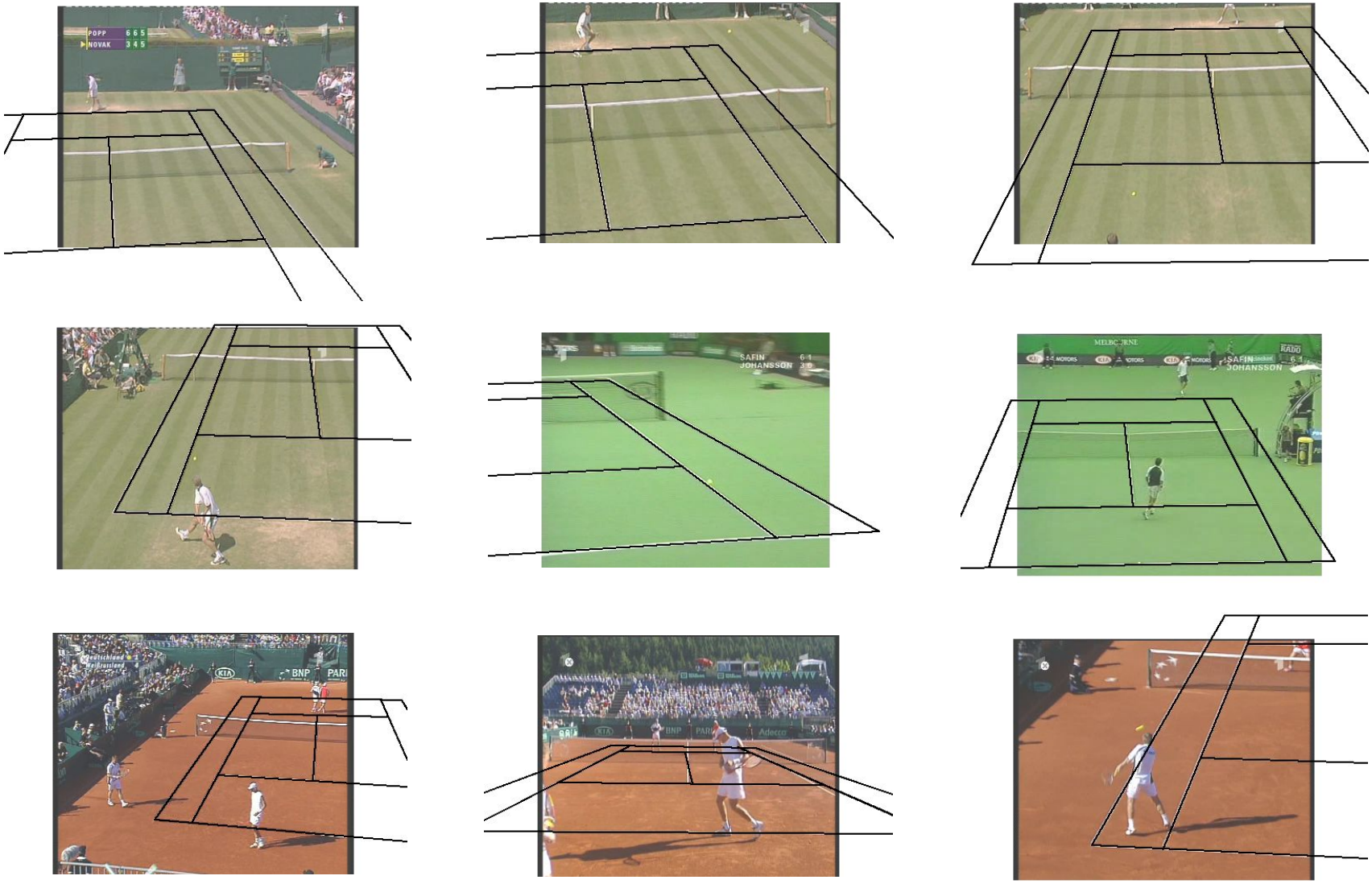


- very large occlusion:

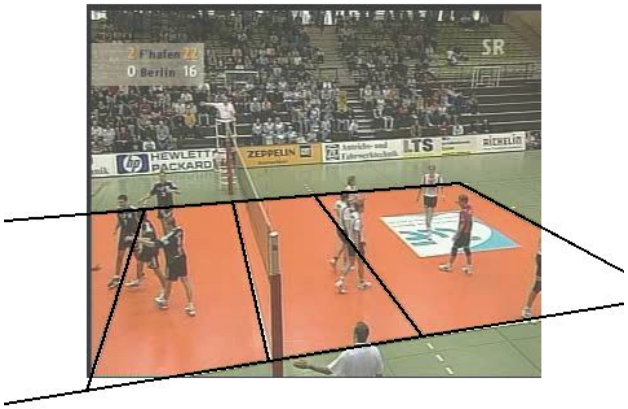


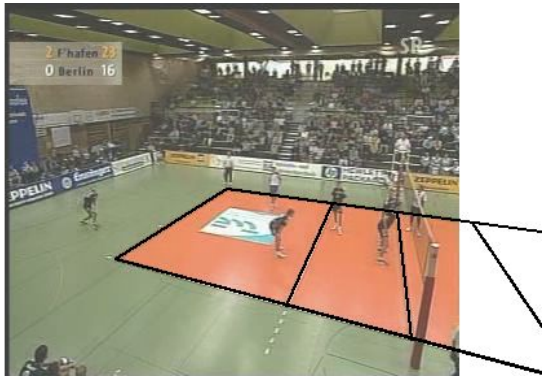
All results were obtained without any parameter adjustment.

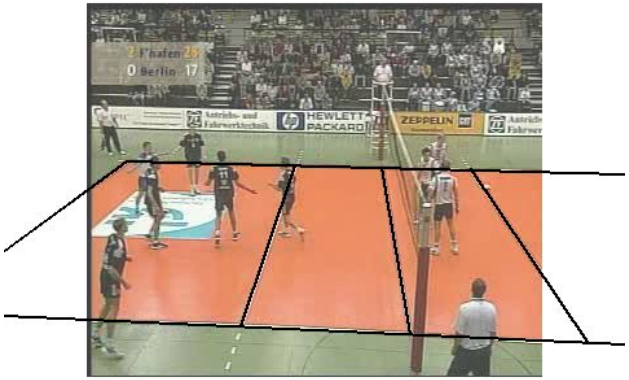
Results (*Tennis*; 2/2)

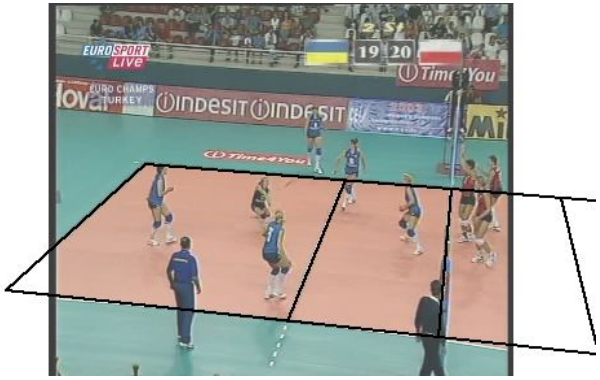


Results (*Volleyball*)

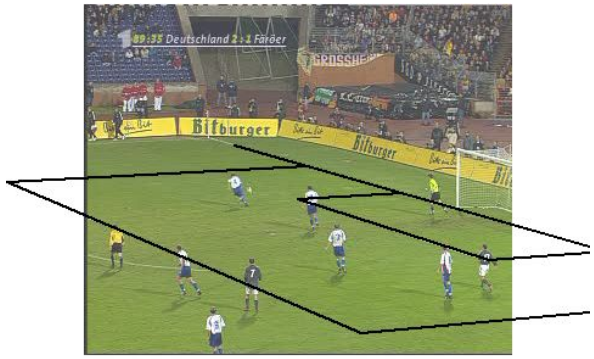


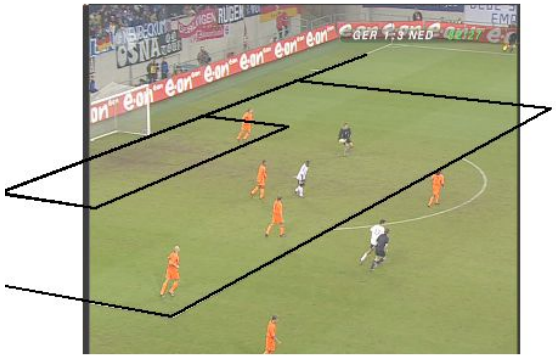


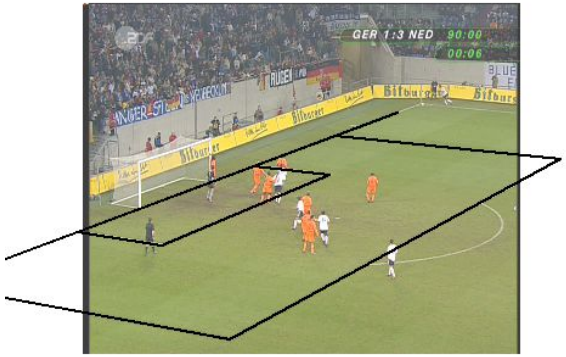


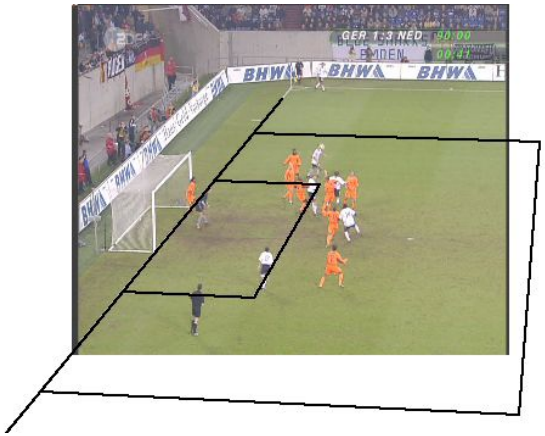


Results (*Soccer*)









Conclusions

- Camera calibration required to convert between screen coordinates and real-world coordinates.

- Generic algorithm, applicable to any court sport:
  - extract line segments of the input image,
  - use model of court-line arrangement,
  - establish correspondences between input lines and model lines.

- Very robust calibration results
  - insensitive to large occlusions,
  - no need for parameter adaptation.

- Computation time
  - 20-35ms for initialization in first frame
  - 4-10ms for tracking (not presented here)
  - [2.8 Ghz Pentium-4, CIF resolution]