Match Video Annotation for Tactical Analytics of Badminton

Dr. Tsì-Uí İk, Professor
Network Optimization Lab (NOL)
Department of Computer Science
National Yang Ming Chiao Tung University

Email: tik@nycu.edu.tw
Web: https://people.cs.nycu.edu.tw/~yi

Competitions

- AI CUP 2023: “Teaching Computer to Watch Badminton Matches”
  - https://aidea-web.tw/topic/cbea66cc-a993-4be8-933d-1aa9779001f8?lang=en
  - Registration was closed on May 2, 2003
  - 250 teams and 439 participants in total.
- IJCAI 2023: “CoachAI Badminton Challenge”
  - https://sites.google.com/view/coachai-challenge-2023/home?authuser=0
  - Registration deadline: May 10th, 2023

Open Source and Open Dataset

- TrackNet
  - GitLab project: https://nol.cs.nctu.edu.tw:234/open-source/TrackNet
  - Tennis dataset: https://nycu1-my.sharepoint.com/:u:/g/personal/tik_m365_nycu_edu_tw/ETCr6-M0e1VDhGCdMbvljcsBe9XyqprS_QgVGD747gzx7A?e=k91F1n
  - Reference: “TrackNet: A Deep Learning Network for Tracking High-speed and Tiny Objects in Sports Applications”
- TrackNetV2
  - GitLab project: https://nol.cs.nctu.edu.tw:234/open-source/TrackNetv2
  - Badminton dataset: https://nycu1-my.sharepoint.com/:u:/g/personal/tik_m365_nycu_edu_tw/EWisYhAiai9Ju7L-tQp0ykEBCSXCSLLimwagy0ziiCRTNQ?e=kWSmnl
  - Reference: “TrackNetV2: Efficient Shuttlecock Tracking Network

Outline

- Part 1: Badminton Match Video Annotation
- Part 2: Camera Geometry
- Part 3: Object Detection and Tracking
- Part 4: Implementation Tutorial

This section will introduce our research work on tactical analysis of badminton based on shot-by-shot data. Firstly, we will introduce the framework of shot-by-shot technical data collection for badminton matches [14]. Secondly, we will present the computer-aided S2-labeling tool developed for singles [15]. Then, some case studies of data analysis and visualization will be given, and shot-by-shot datasets will be shared for academic research. Finally, we will introduce an AI challenge in AI CUP supported the Ministry of Education, Taiwan [23].

PART 1: BADMINTON MATCH VIDEO ANNOTATION

Microscopic vs Macroscopic

Frame

- Frame-level data: shuttlecock trajectory, locations and skeletons of players
- Key tasks: TrackNet, YOLO, OpenPose/VIBE/MediaPipe

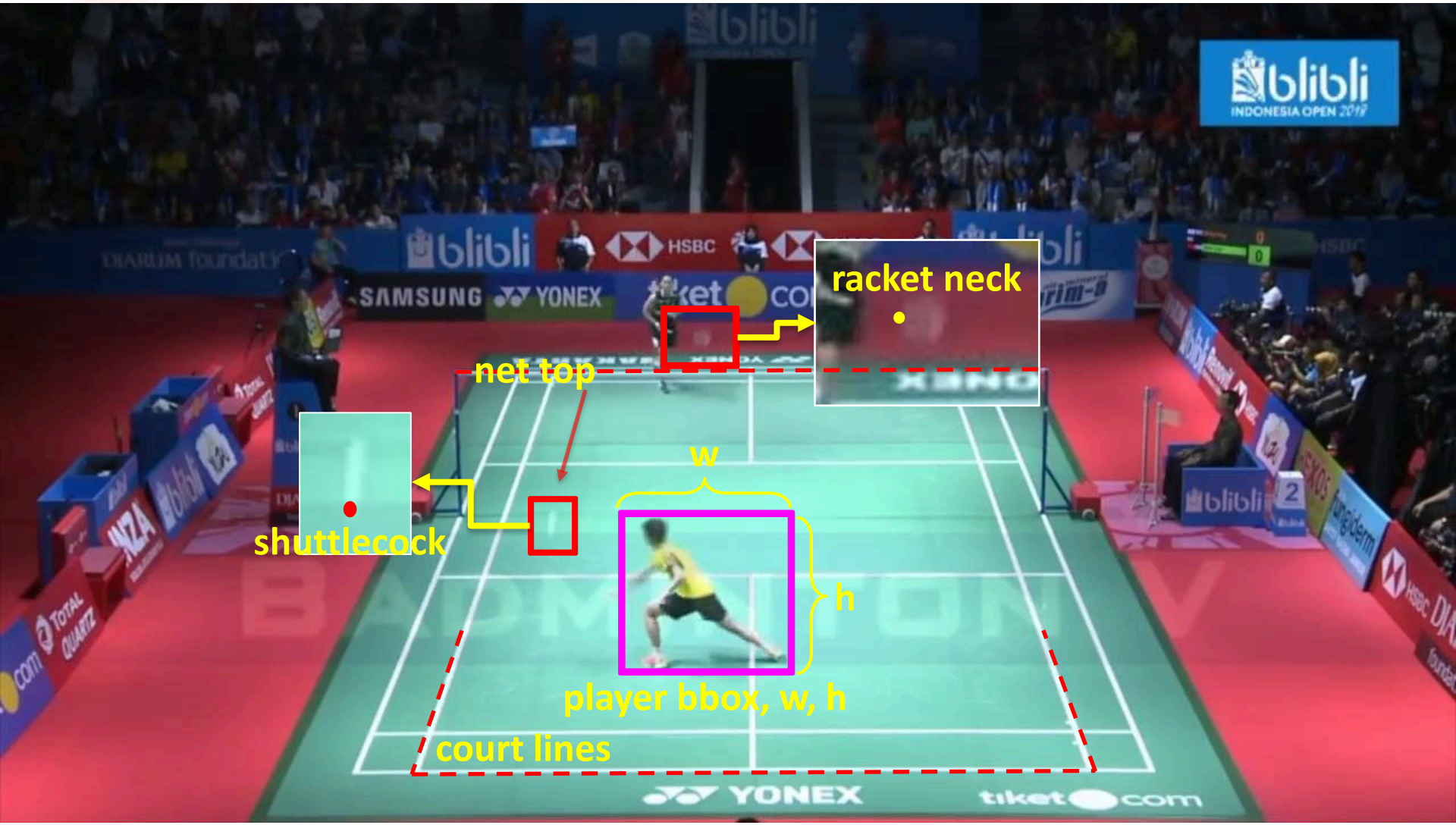
Rally

- Rally-level data: serve/return/dead bird, hitting time, ball type, win/loss
- Key tasks: rally segmentation, shot event detection, ball type classification, win/loss judgement, …

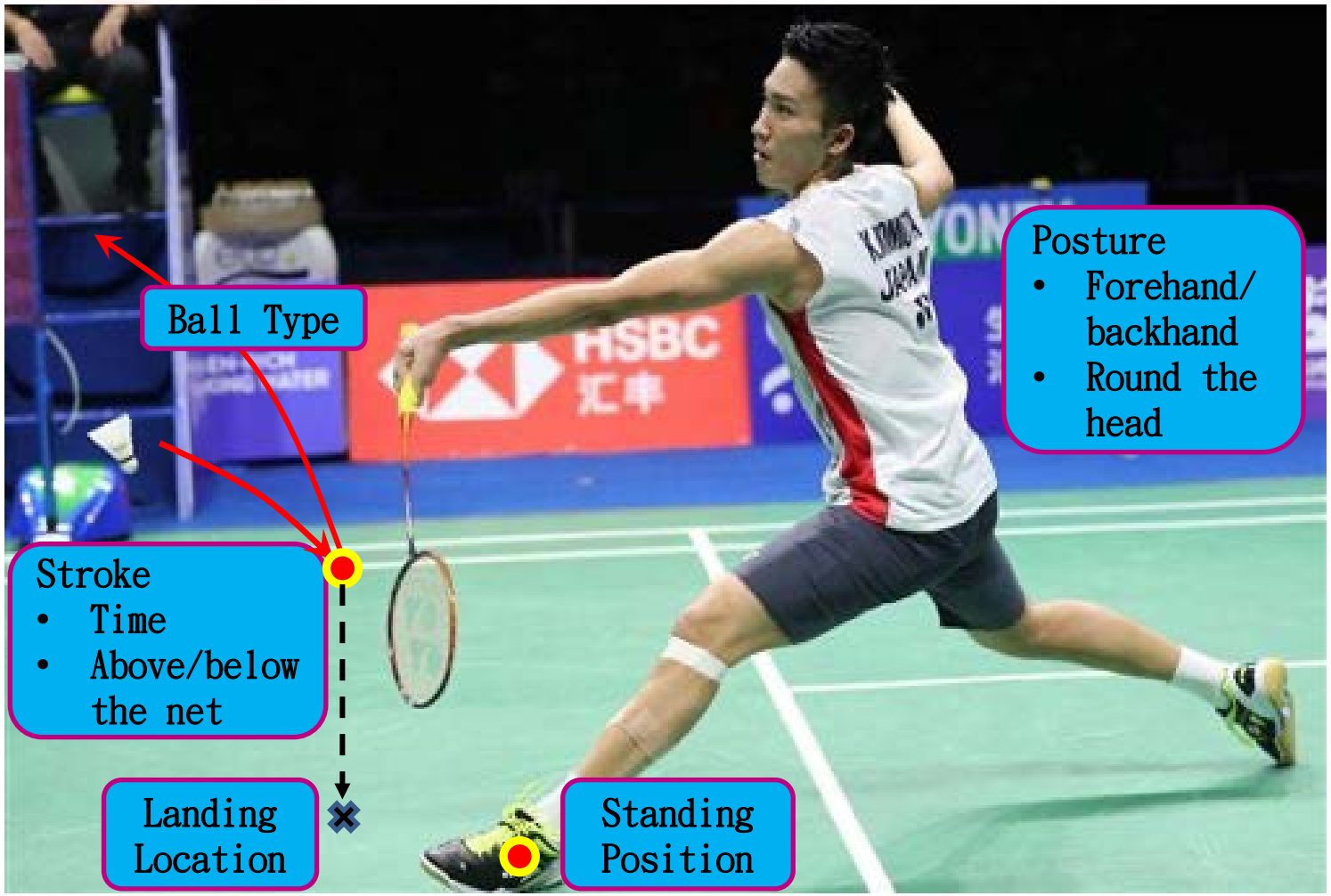
Match

- Match-level information: singles/doubles, players, winner, video, …
- Set-level information: serve/side, score, …
- Key tasks: database design

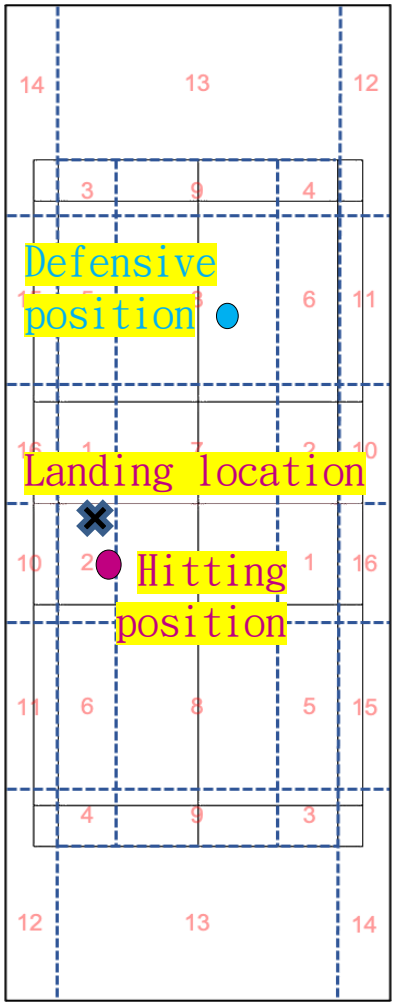
Video Tells Stories



Microscopic Data



Serve/Return/……/Dead Bird



Court Layout

Shot-by-Shot Technical Data

The shot-by-shot (S^2) technical data record microscopic data of each shot including temporal, spatial, posture, and skill performance from the serve to the dead bird.

	player	server	start	player location	Opponent location	type	aroundhead	backhand	hit height	landing area	lose reason	win reason	shot winner	flaw
1	A	1	0	8 [714,454]	8 [598,585]	發短球		1	2	7 [654,543]				
2	B	2	40	8 [644,578]	8 [707,451]	推球		1	2	4 [786,424]				
3	A	2	73	8 [774,425]	8 [698,596]	殺球			1	5 [875,587]				
4	B	3	87	8 [720,612]	4 [795,421]	擋小球			2	Out [863,454]	出界		A	

Player A: Serve

Player B: Push

Player A: Smash

Player B: Block shot

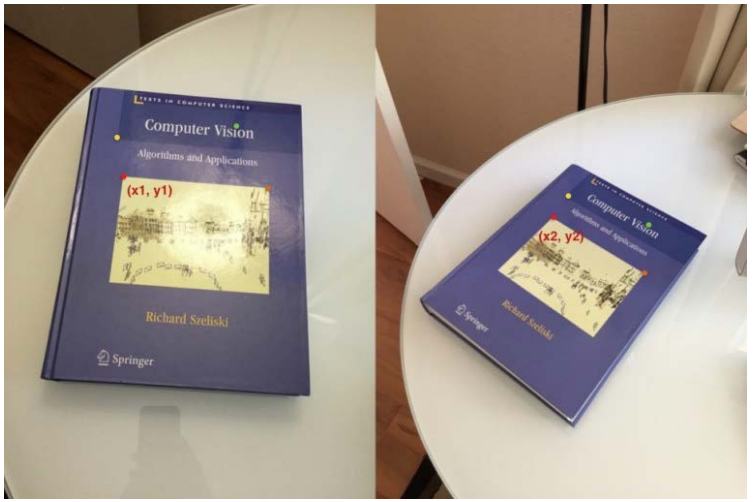
Attacking player Defensive player Trajectory tail Trajectory head

Shot-by-Shot Technical Data

- Temporal data: the hitting time of each shot from the serve to dead bird
- Spatial data: standing position of players, height and location of hitting point
- Posture data: forehand/backhand, round the head
- Skill data: shot event type, shot player, serving ball type, return ball type, win or loss

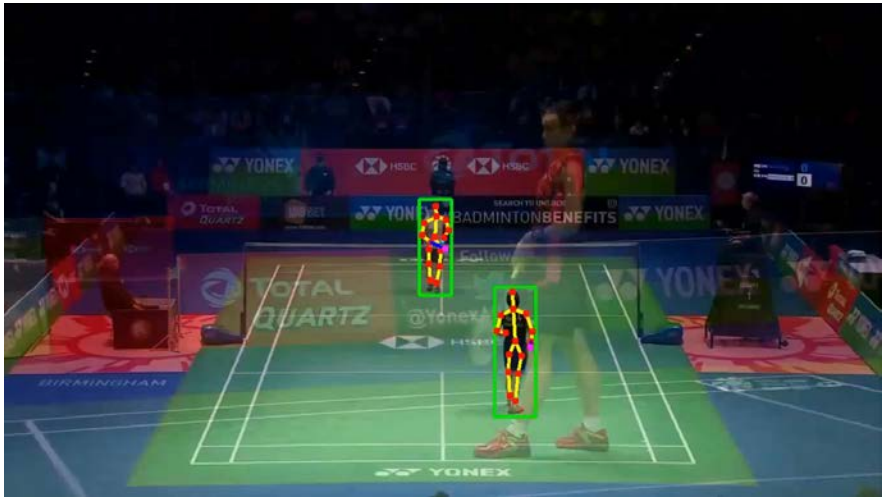
Computer Vision Aided

TrackNet ➤

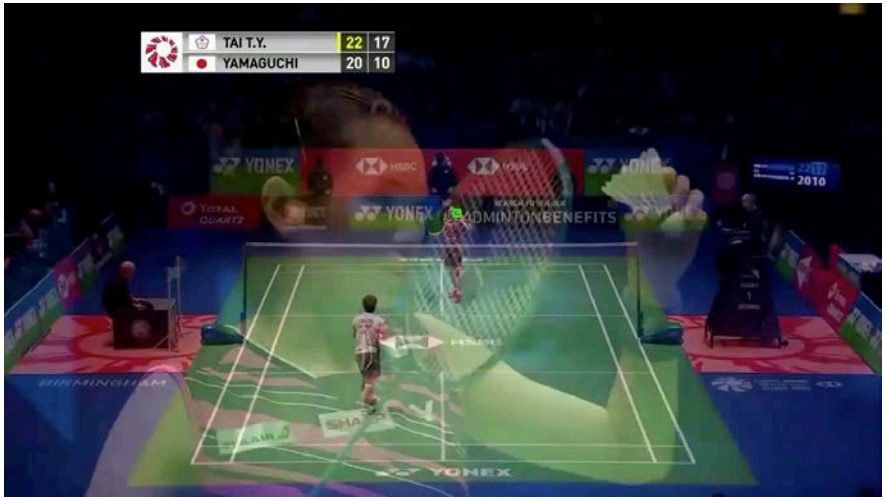


➤ Homography

TrackNet + YOLOv3 + OpenPose ➤



➤ Shot Detection



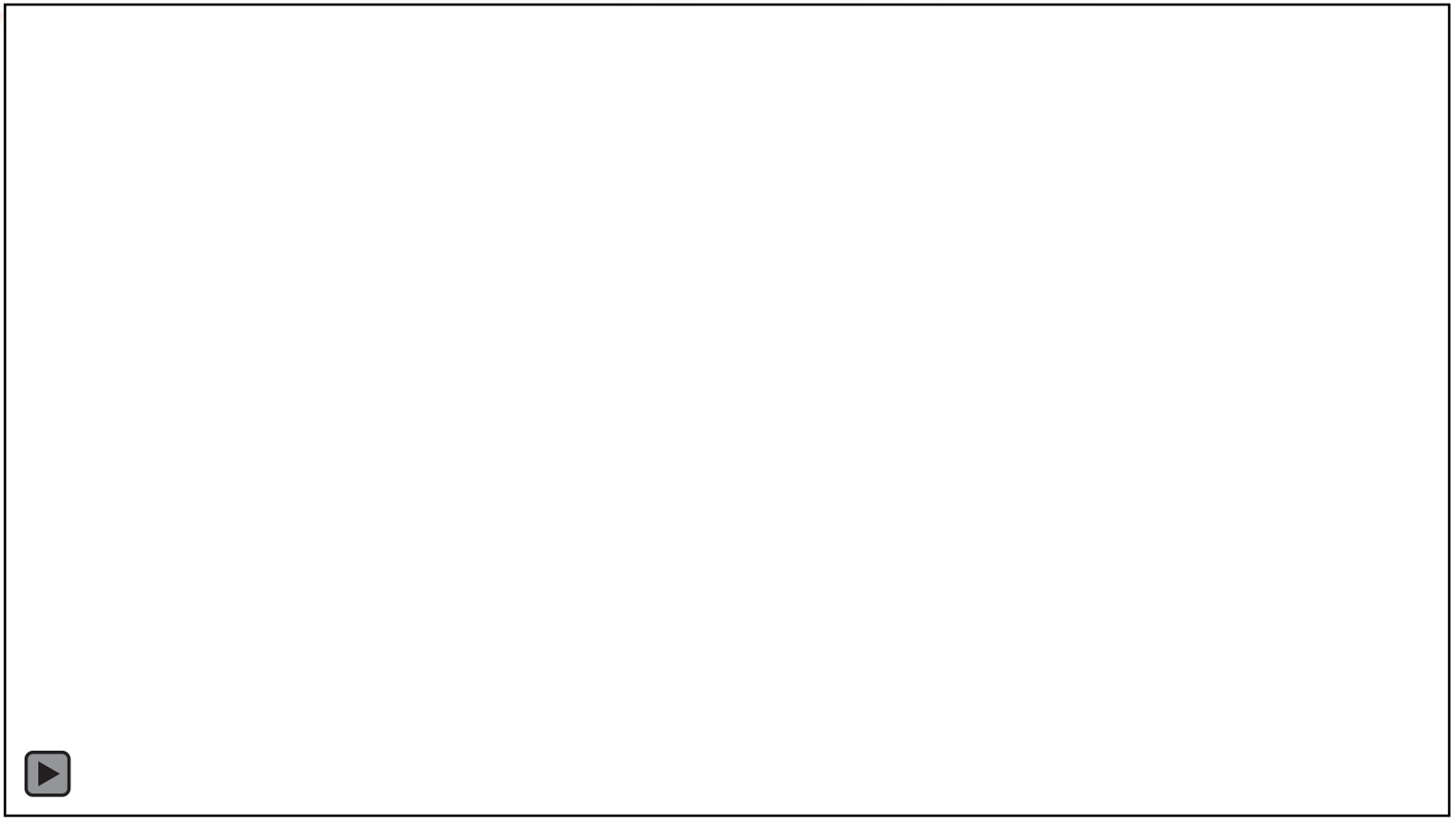
serve hit other ground

S² Labeling Tool

AI CUP Dataset

- The dataset consists of around 1000 rally videos.
  - Each rally videos recorded one complete rally, from the serve to the dead bird, companioned with a shot-by-shot tagging file.
    - The average length of a rally video is 13 seconds.
    - In average, a rally consists of 13 shots.
  - The rally videos are encoded in mp4, and the tagging files are in csv format.
  - The footage was taken from the back-top view. In each clip, the focal length and view angle of the camera are fixed.
  - Frame rate: 30 FPS; resolution: 1280x720; bitrate: 2500 kbps

Example: Serving Miss



ball_round	start_frame	player	server	type	aroundhead	backhand	hit_height	landing_x	landing_y	player_location_x	player_location_y	opponent_location_x	opponent_location_y	getpoint_player
1	40832	B	3	發長球			2	712	366	618	586	706	410	A

Frequent issues: 1. The last shot; 2. Occluded; 3. Jump

Example: Multiple Shots

ball_round	start_frame	player	server	type	aroundhead	backhand	hit_height	landing_x	landing_y	player_location_x	player_location_y	opponent_location_x	opponent_location_y	getpoint_player
1	25744	A	1	發長球			2	781	379	626	565	722	405	
2	25805	B	2	點扣			1	453	574	770	380	687	596	
3	25828	A	2	擋小球		1	2	582	457	524	576	706	395	
4	25853	B	3	撲球			1	519	645	598	436	480	581	B

Hitting Time (Temportal)

Normal

Occluded

W/O Hitting Frame

Standing Position (Spatial)

Normal

Normal

Occluded

Jump

Height of Hitting Point (Spatial)

Lower than the net

Higher than the net

Projection of Hitting Location (Spatial)

Sideview

Normal

Occluded

Posture Tags

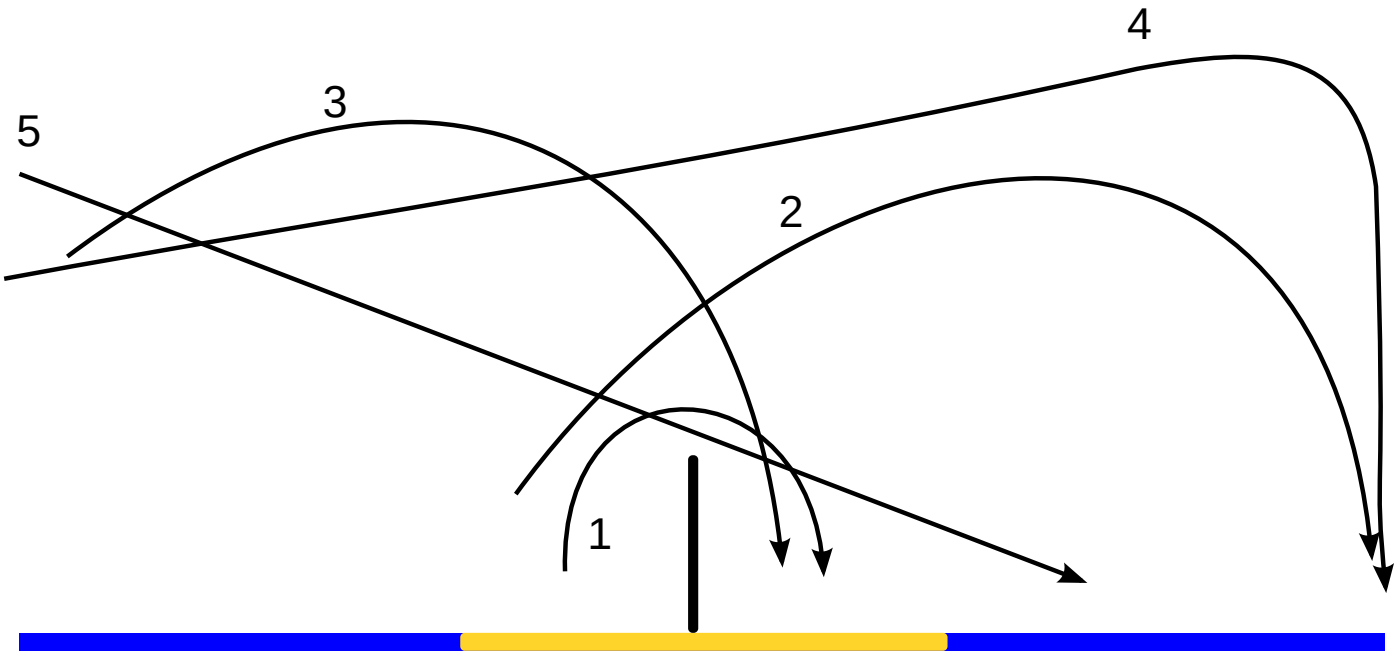

Forehand


Round the head


Backhand

Not round the head

Ball Types



Basic ball type: 1. Netplay; 2. Lob 3. Drop 4. Long 5. Smash

Gera 46, CC BY 3.0, https://commons.wikimedia.org/w/index.php?curid=6826284

Statistics Data (1)

Fig1. Ball types usage

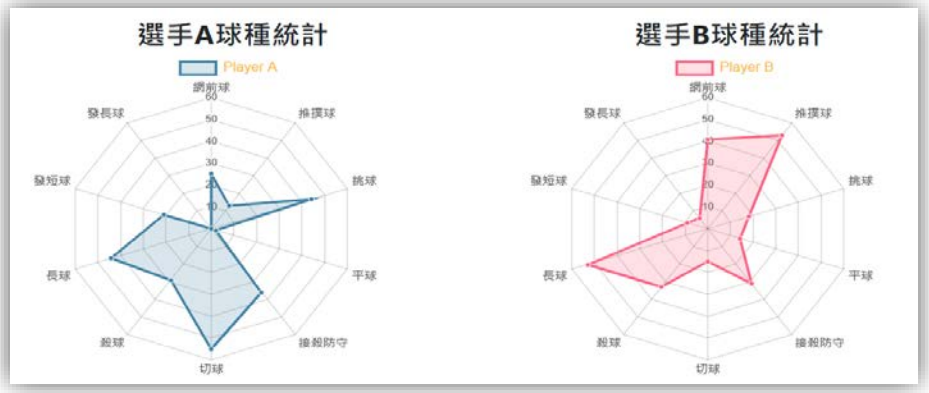


Fig3. Ball types of winning shots

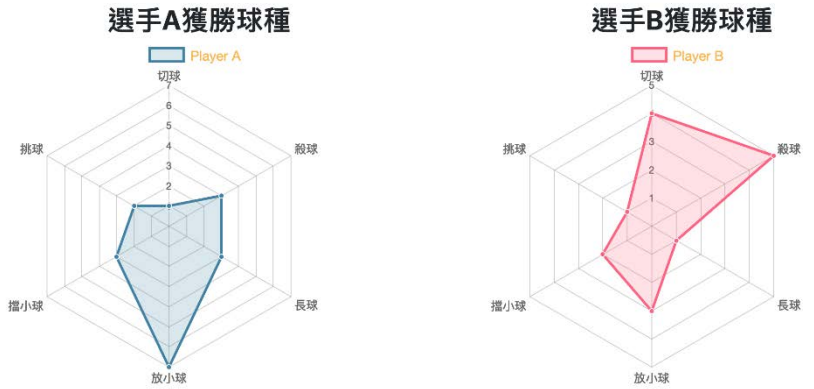


Fig2. #shots to win a rally

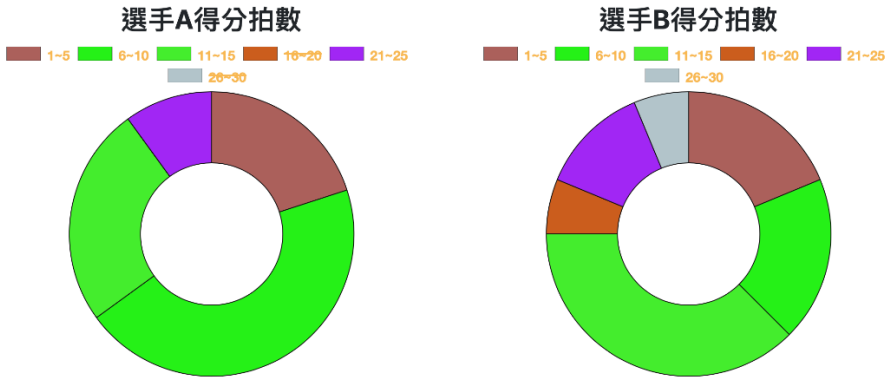


Fig4. Ball type pattern

- Frequent patterns of the last three shots
  1. C Push - M Smash - C Block
  2. C Long - M Smash - C Block
  3. C Lob - M Smash - C Block
- Frequent patterns of three shots
  1. C Long - M Smash - C Block
  2. C Push - M Smash - C Block
  3. C Smash - M Block - C Netpaly

Statistics Data (2)

Fig4. Reason of loss

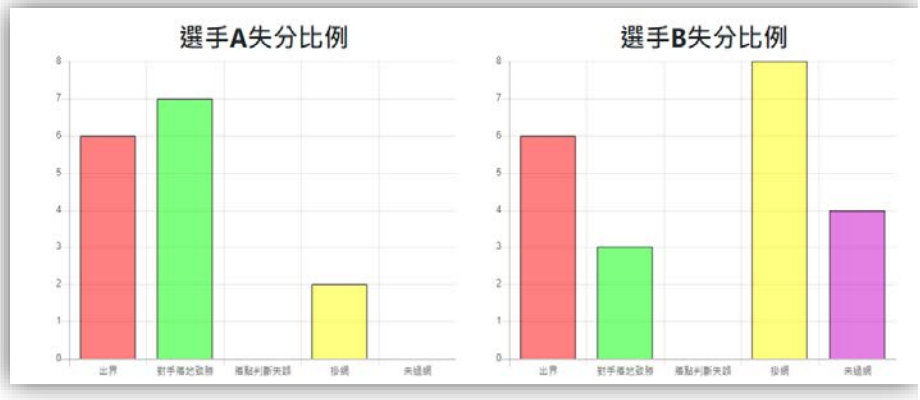


Fig5. Types of loss

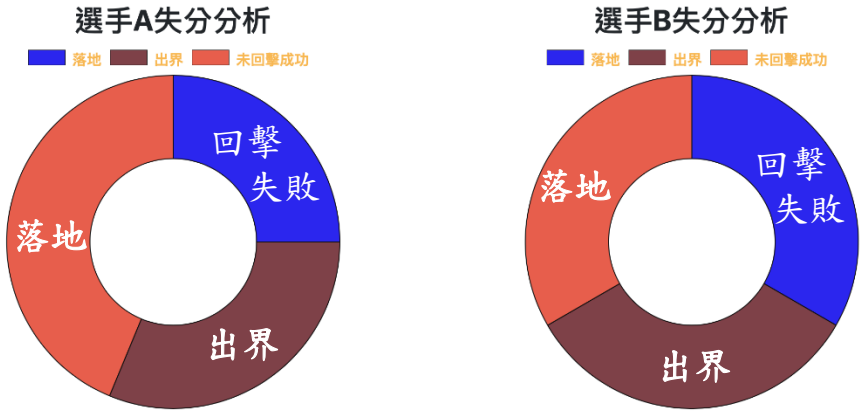
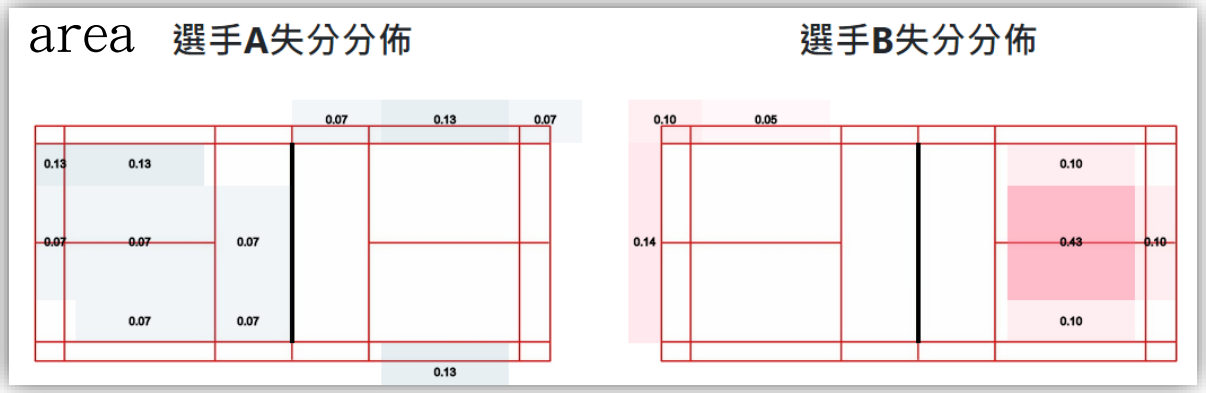


Fig6. Distribution of points lost by court area



Fuzhou Open 2019

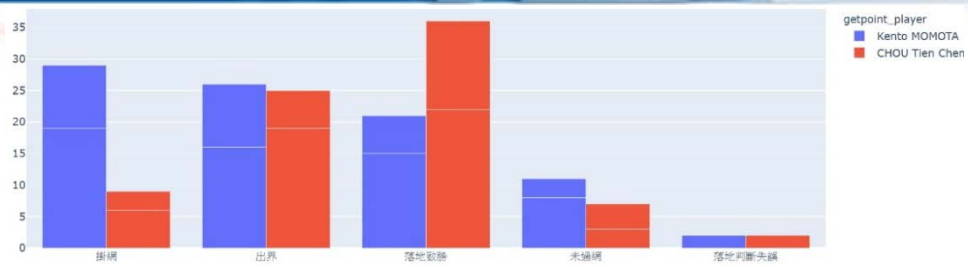
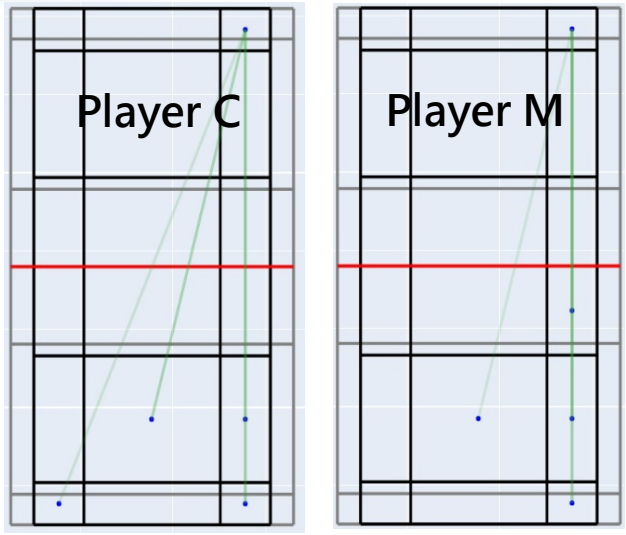


Fig 1. Loss reasons ↑



↓ Fig 2. Effective smash

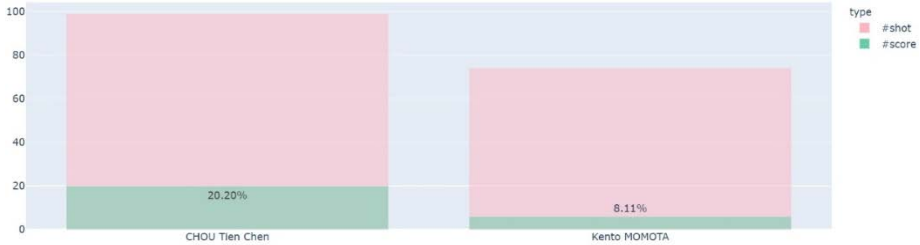


Fig3. Landing location

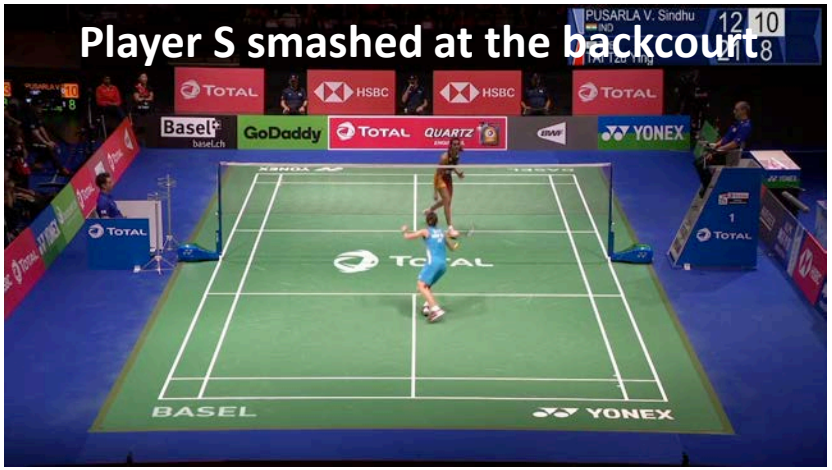
Player M: play at the backcourt

Thematic Video Clips

Player C: play at the backcourt

Thematic Clips

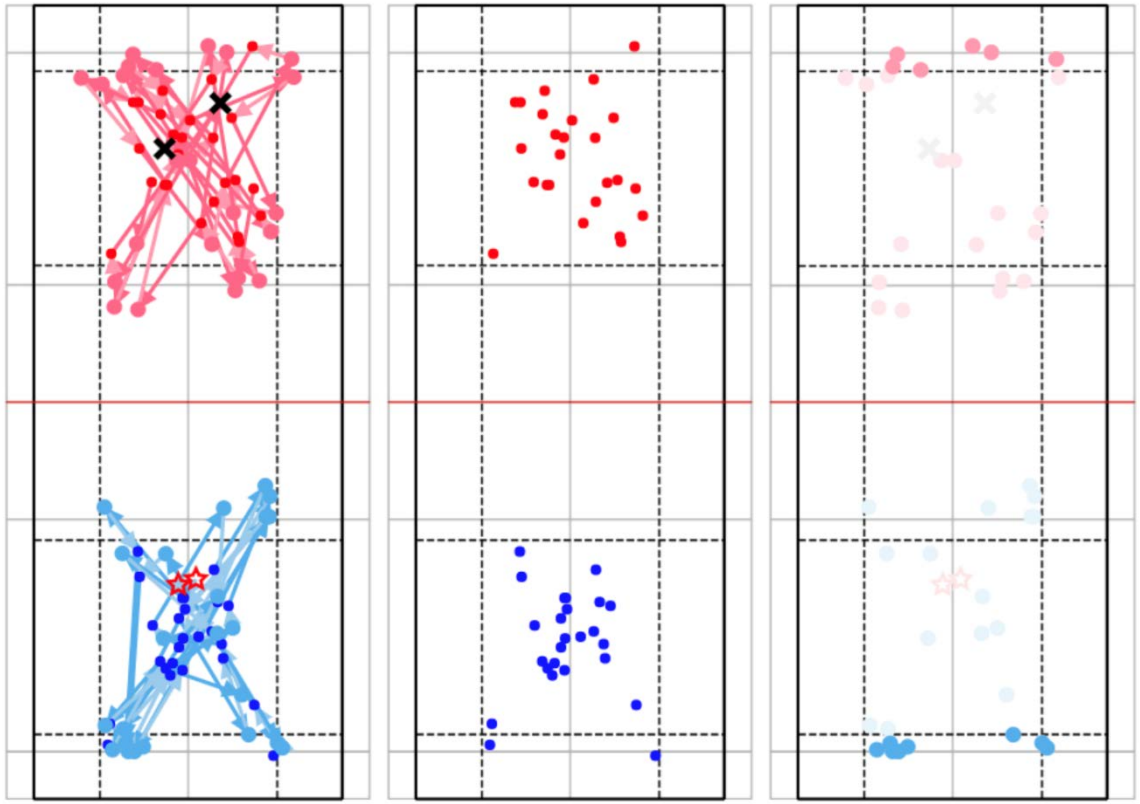
Smash at the Backcourt



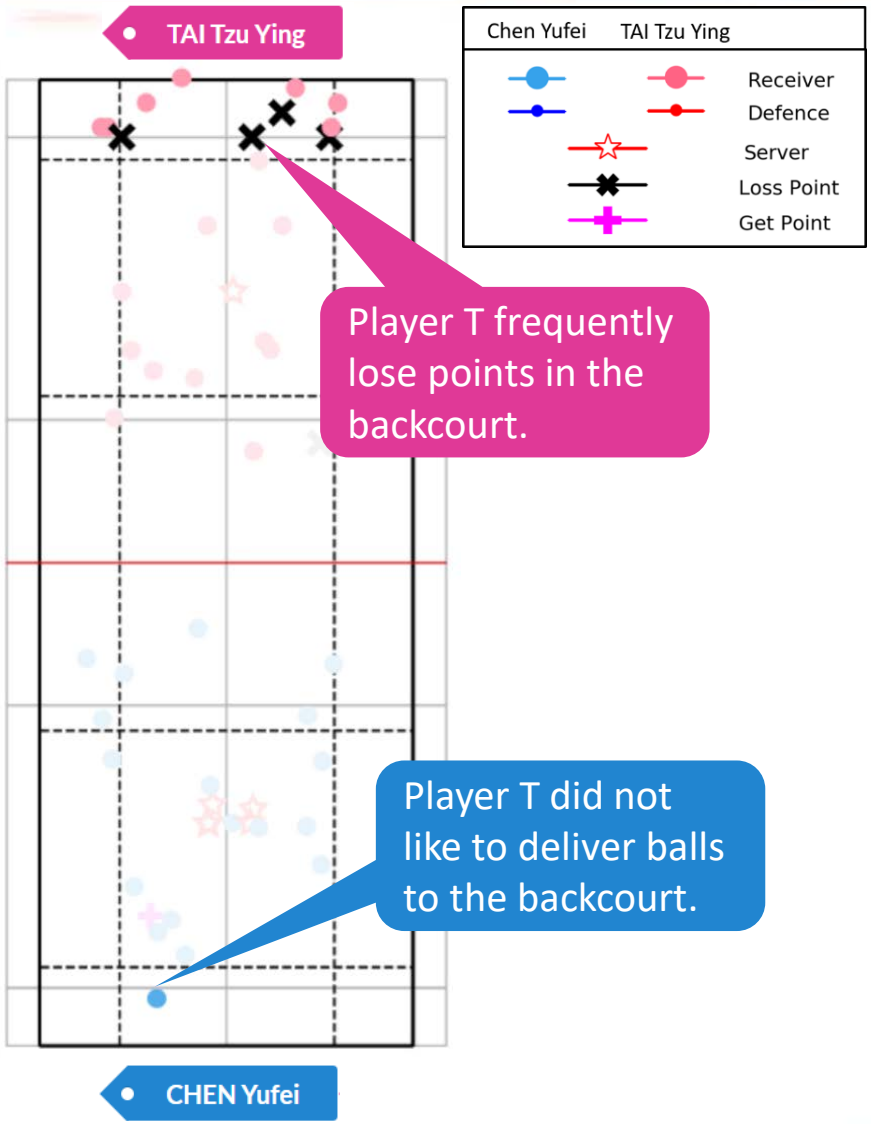
Frequent Pattern of Loss (Long → Smash → Block)

Play Standing and Defensive Standing

Fuzhou Open 2019 – Player M vs. Player C



- Set 1; Rally 27~28
- C: red; M: blue
  - Left: moving pattern
    - Typical X shape
    - C's play standing positions and defensive standing positions are more dispersed than M's.
  - Middle: defensive standing
  - Right: backcourt play standing
    - M's standing is at two corners.
    - C's standing is more evenly distributed.



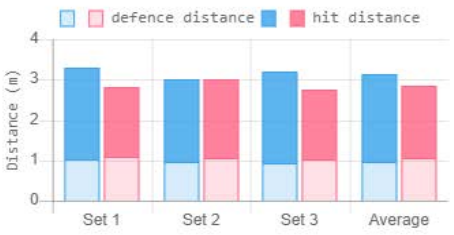
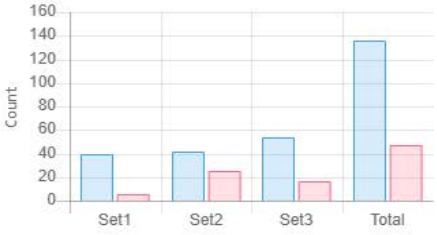
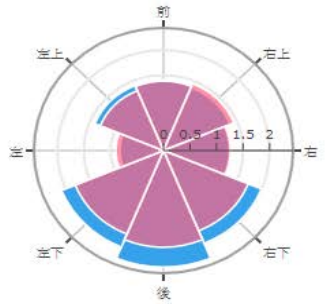
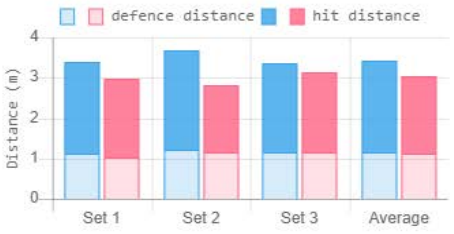
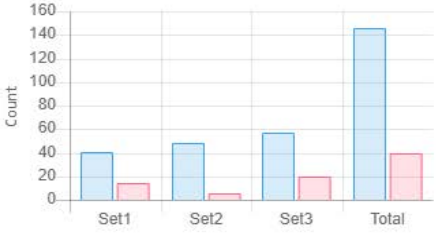
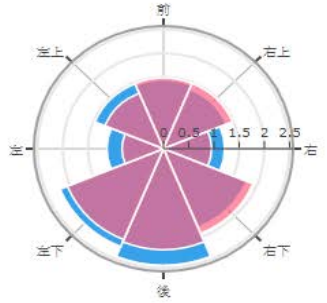
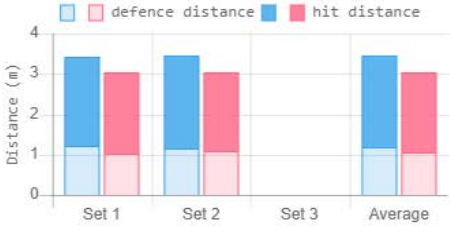
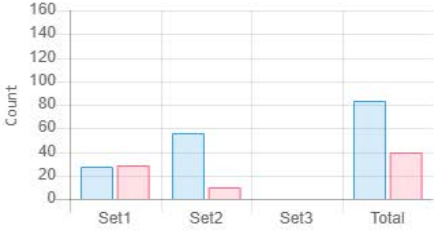
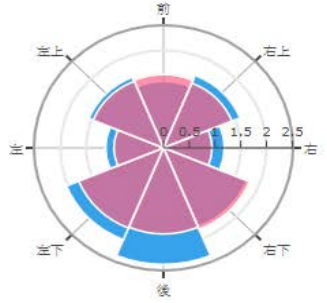
Performance at Backout

Malaysia Masters 2020 Finals
Player C vs. Player T

- Set 2; Rally 24~28
- Player T (red): many backcourt plays with misses
- Player (blue): only 1 backcourt play

Comparison of Multiple Games

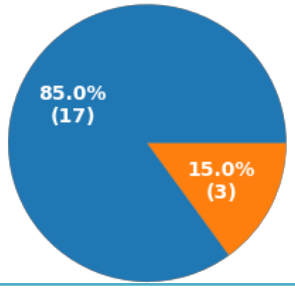
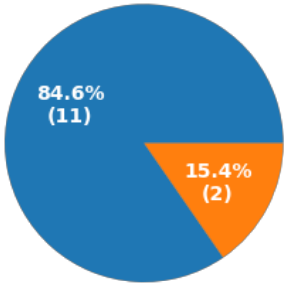
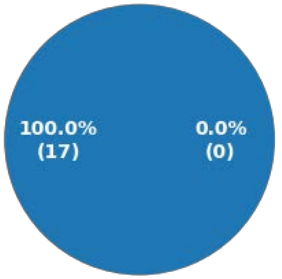
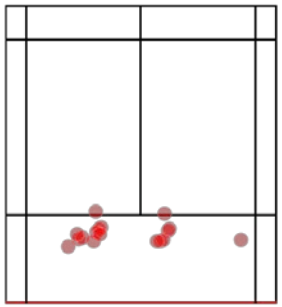
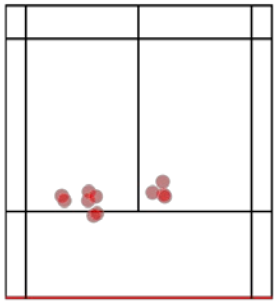
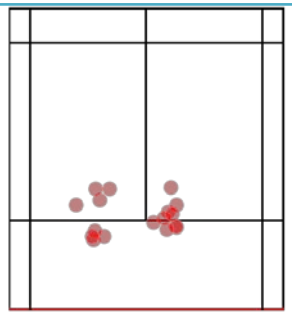
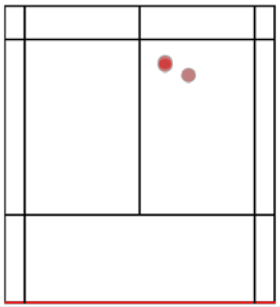
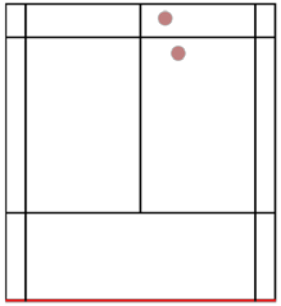
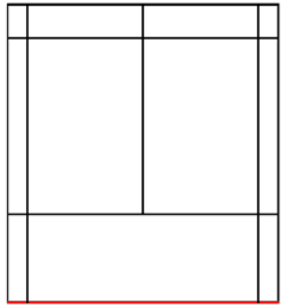
Running Distance | Backcourt Play | Moving Speed

Match Name	Running Distance per Rally	Number of Backcourt Play	Average Velocity by Direction
2019-08-23 World Championships 2019 Quarter-finals TAI Tzu Ying PUSARLA V. Sindhu			
2019-12-15 World Tour Finals 2019 Finals TAI Tzu Ying CHEN Yufei			
2020-01-12 Malaysia Masters 2020 Finals TAI Tzu Ying CHEN Yufei			

Player T was faster than her opponents in most directions.

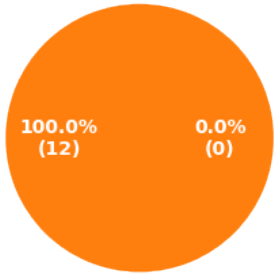
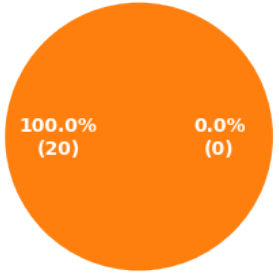
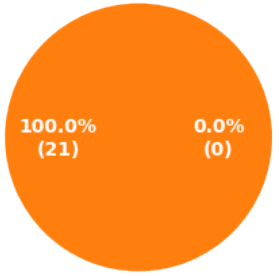
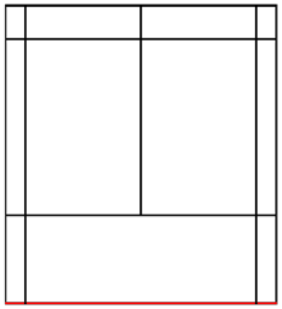
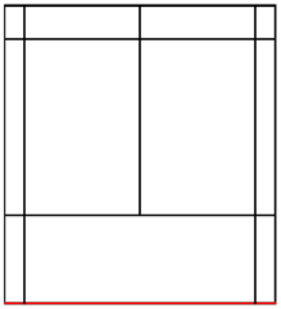
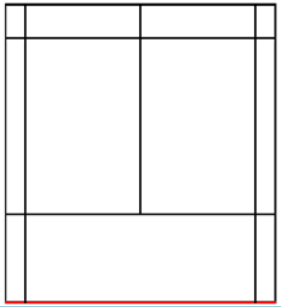
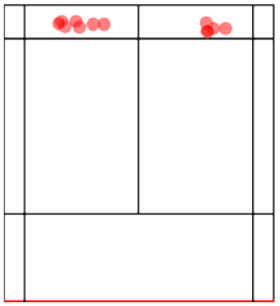
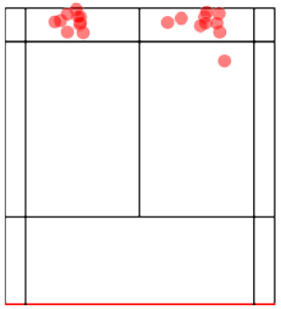
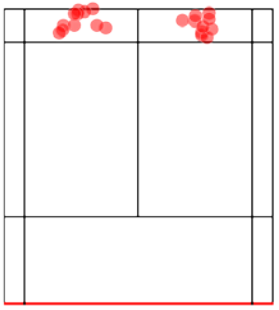
Player T received more at the backcourt than her opponents.

T: Serving Short

World Tour Finals 2019 – Chen Yufei vs. TAI Tzu Ying			
Set #	Set1	Set2	Set3
Serving (% and #)			
Short			
Long			

廖偉喆，& 李光武.（2016）. 世界級女子羽球單打選手發球與接發球技術分析以戴資穎為例. *海峽兩岸體育研究學報*, *10*(1), 75-84.

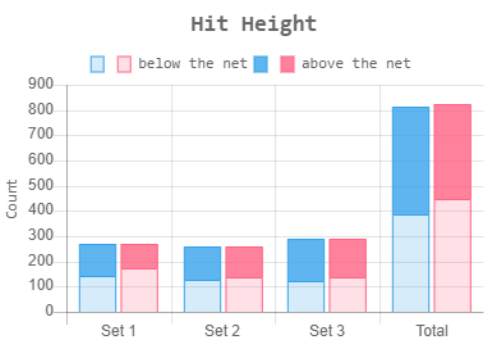
C: Serving Long

World Tour Finals 2019 – Chen Yufei vs. TAI Tzu Ying			
Set #	Set1	Set2	Set3
Serving (% and #) Ball Type: short, long			
Short			
Long			

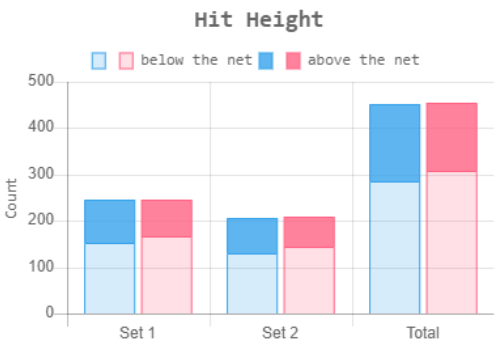
廖偉喆，& 李光武. (2016). 世界級女子羽球單打選手發球與接發球技術分析以戴資穎為例. *海峽兩岸體育研究學報*, *10*(1), 75-84.

Advantage vs. Disadvantage

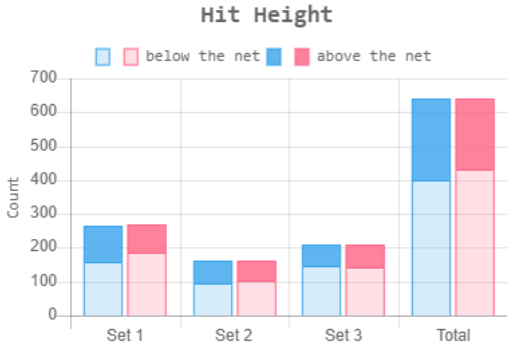
C (red)
vs.
M (blue)



Fuzhou 2019

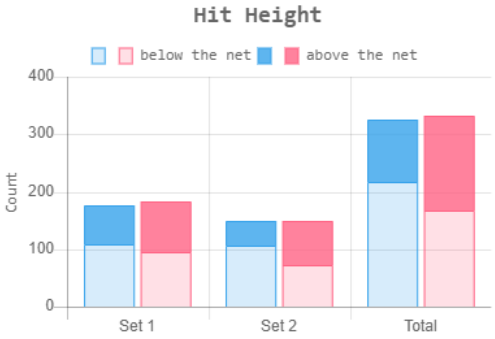


Korea 2019

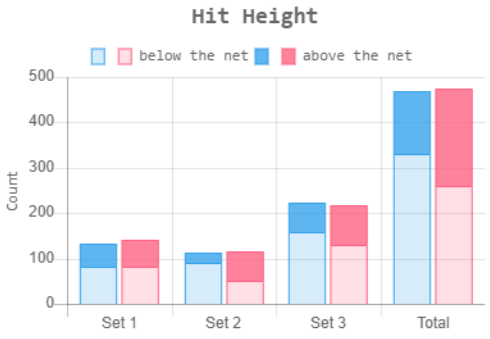


Denmark 2018

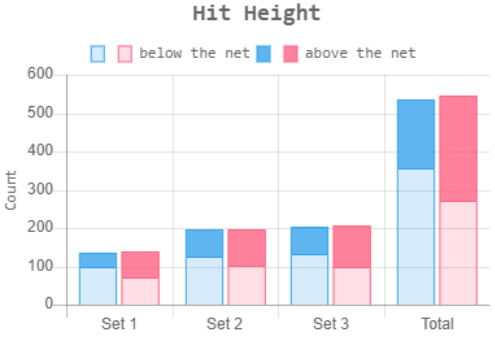
T (red)
vs.
C (1)(2)
S (3)



(1) Malaysia Masters 2020



(2) World Tour Finals 2019



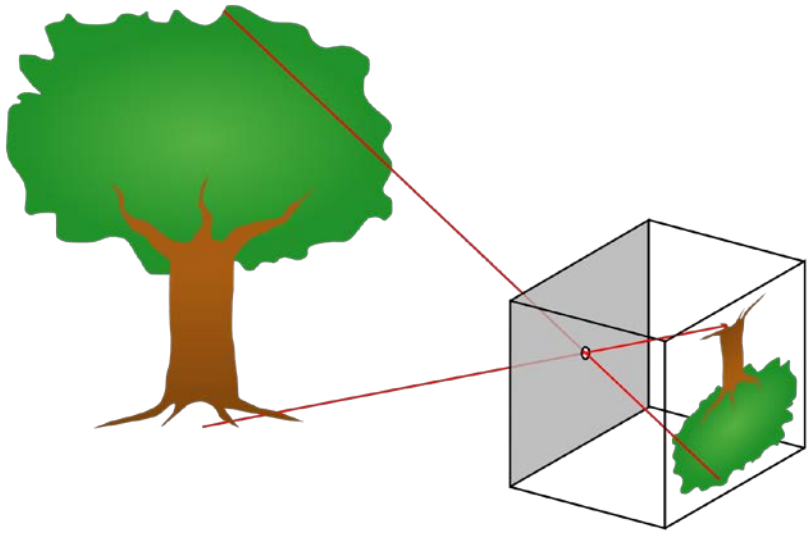
(3) World Championships 2019

This section will introduce the geometric foundations of computer vision, including the pinhole imaging principle, internal and external parameters of cameras [4], lens distortion correction [5], 2D homography mapping [22], 3D positioning [6,7], and camera pose estimation. These are the fundamental techniques for performing geometric calculations through imaging.

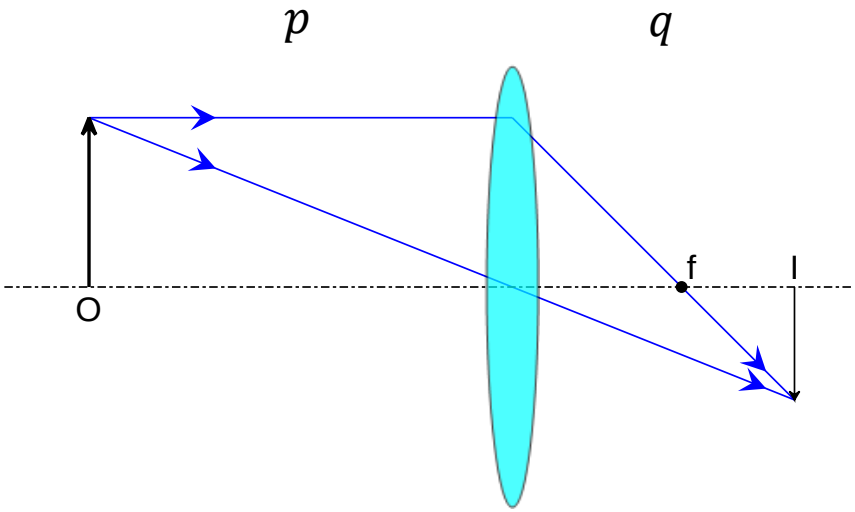
PART 2: CAMERA GEOMETRY

Imaging Models

Pinhole Imaging

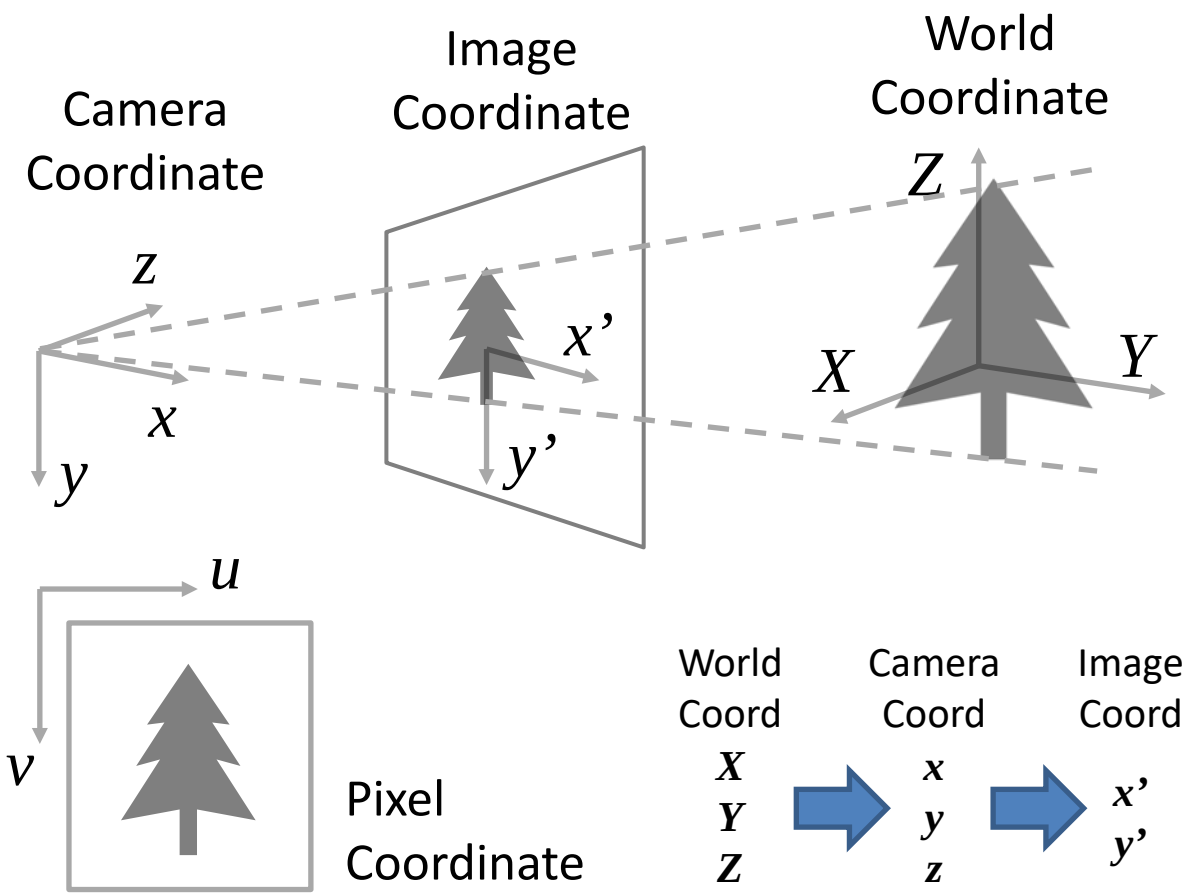


Convex Lens Imaging



$$\frac{1}{p}+\frac{1}{q}=\frac{1}{f}$$

Coordinate Systems



2D Homogeneous coordinate

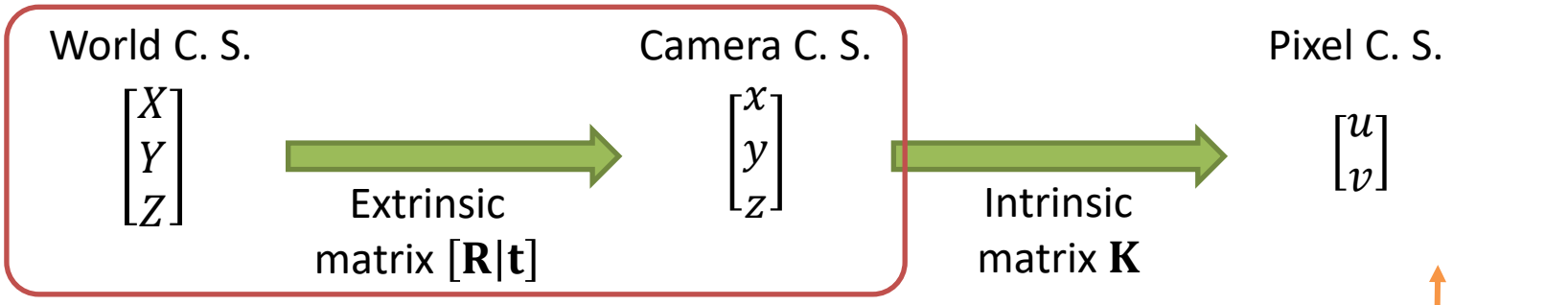
$$(u, v) \rightarrow \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

3D Homogeneous coordinate

$$(X, Y, Z) \rightarrow \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

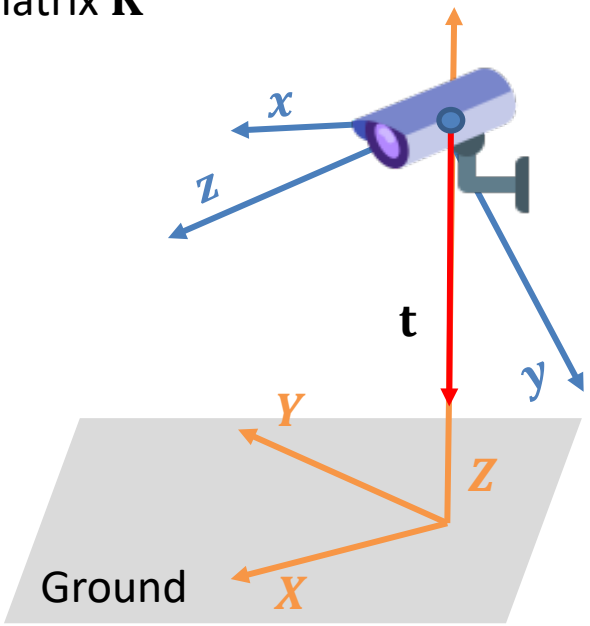
World Coord	Camera Coord	Image Coord	Pixel Coord
X Y Z	x y z	x' y'	u v

Extrinsic Matrix



Rotation Translation

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = [\mathbf{R}|\mathbf{t}] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{21} & r_{31} & t_x \\ r_{12} & r_{22} & r_{32} & t_y \\ r_{13} & r_{23} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$



Roadside CCTV as an example

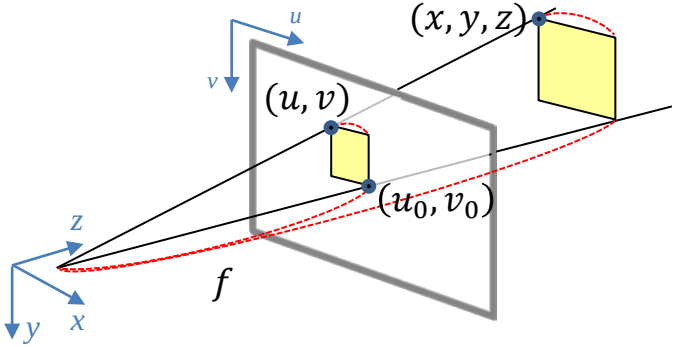
$$[\mathbf{t}]_C = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} = \mathbf{R} \begin{bmatrix} 0 \\ 0 \\ -h \end{bmatrix}$$

where h is the height of camera.

Intrinsic Matrix

- f: the focal length
- The relation between the camera coordinate and the image coordinate

$$z\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

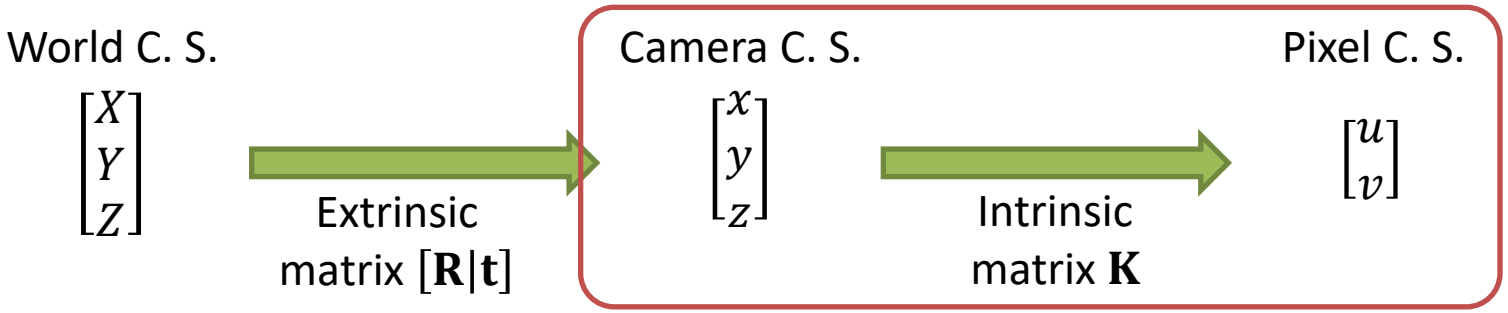


$$\frac{u-u_0}{f} = \frac{x}{z}$$
$$\frac{v-v_0}{f} = \frac{y}{z}$$

- f_u, f_v: the focal length (in the unit of pixels)
- u_o, v_0: the image center in the pixel coordinate
- The relation between the camera coordinate and the pixel coordinate

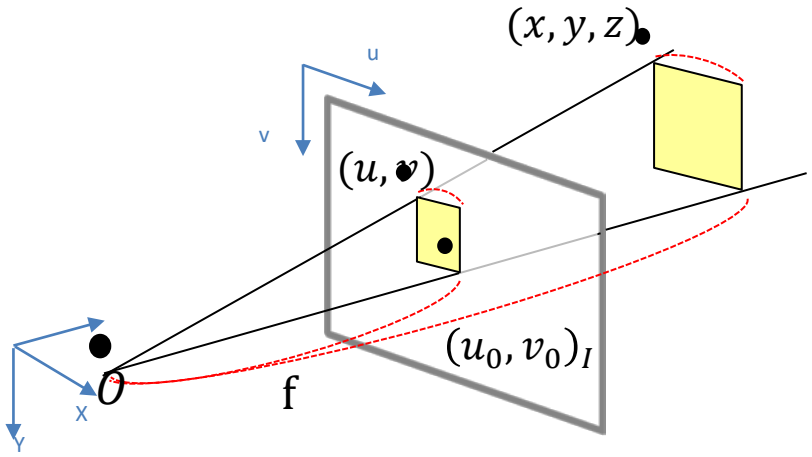
$$z\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Perspective Projection



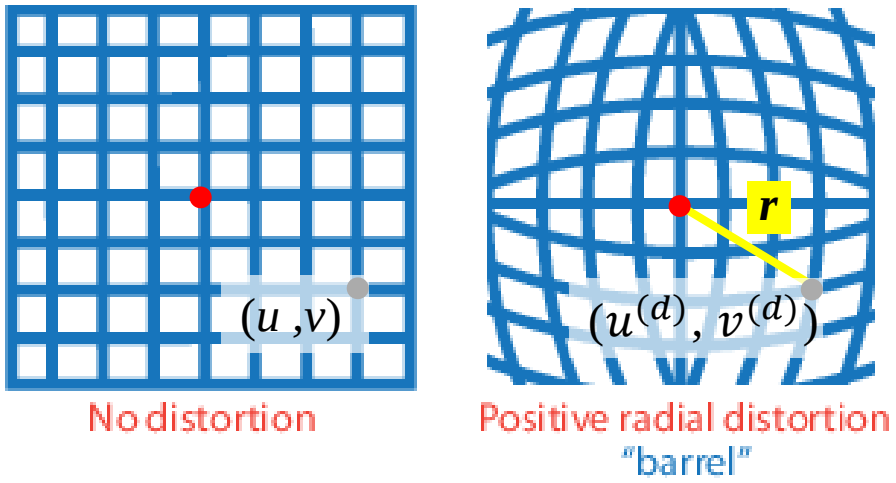
$\mathbf{K} = \begin{bmatrix} f_x & s & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}$, where f_x, f_y is the focal length of the camera, and s is skew. Typically, $f_x = f_y$, $s = 0$.

$$z\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{K}[\mathbf{R}|\mathbf{t}]\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

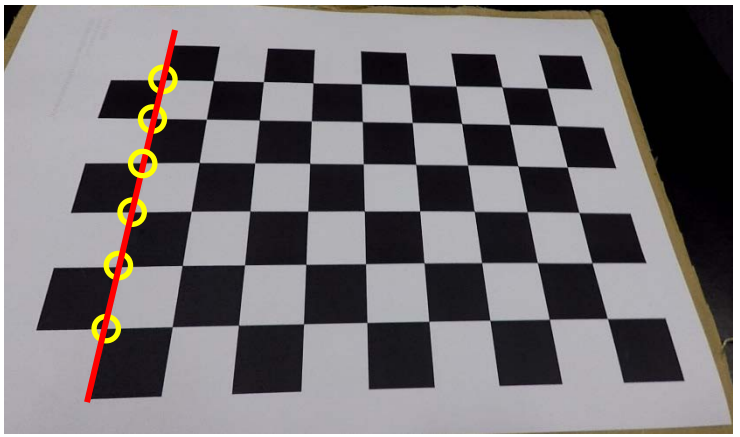


Lens Distortion Removal

"Barrel" Distortion Model



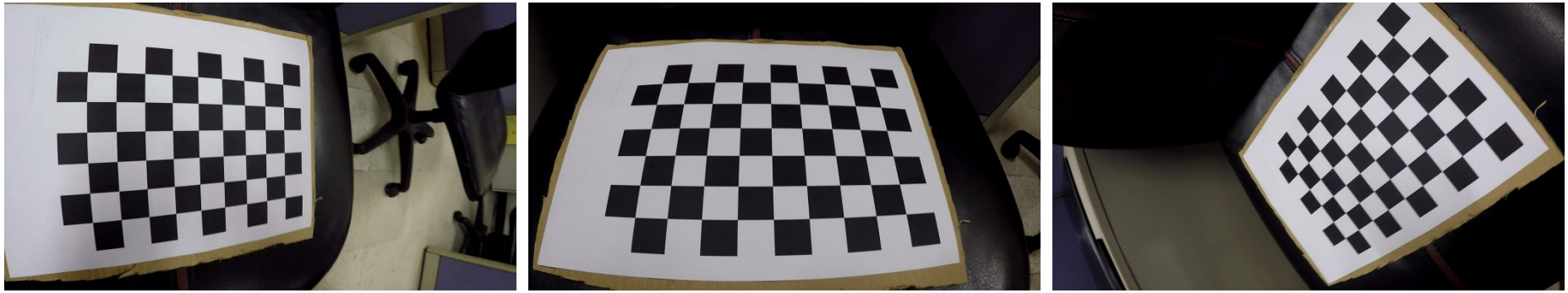
$$r = \sqrt{u^{(d)^2} + v^{(d)^2}}$$
$$u = u^{(d)}(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$v = v^{(d)}(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$



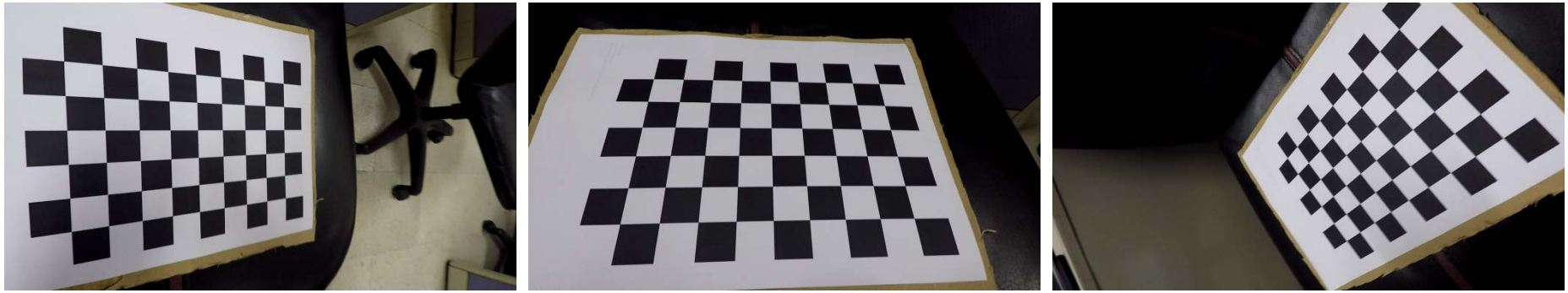
Chessboard Method for
Lens Distortion Removal as well as
Intrinsic Matrix Estimation

Example of Distortion Removal

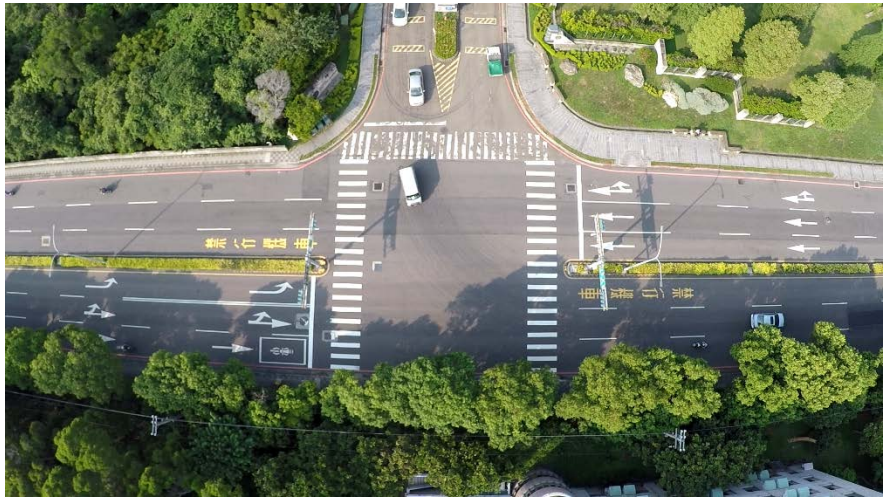
Before Calibration

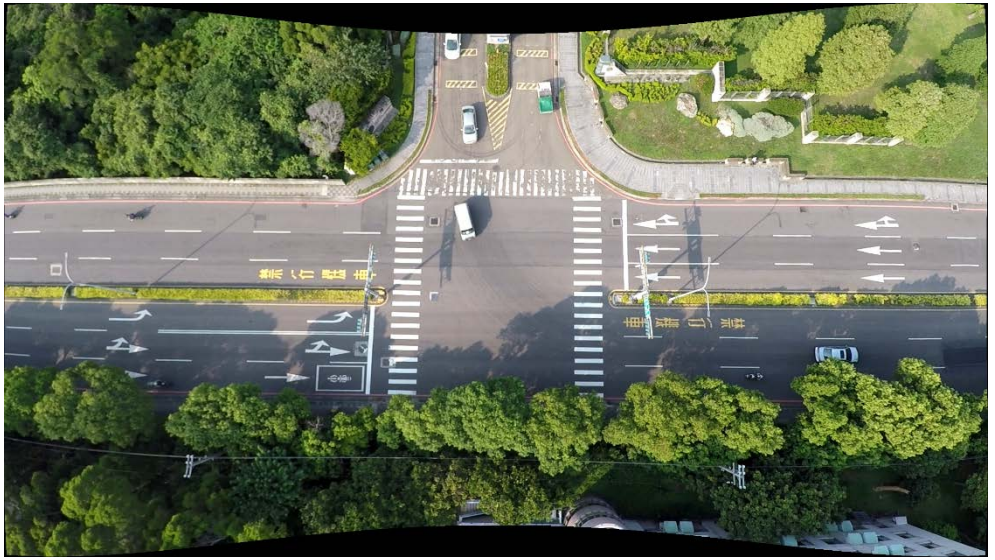


After Calibration



Aerial Image as an Example

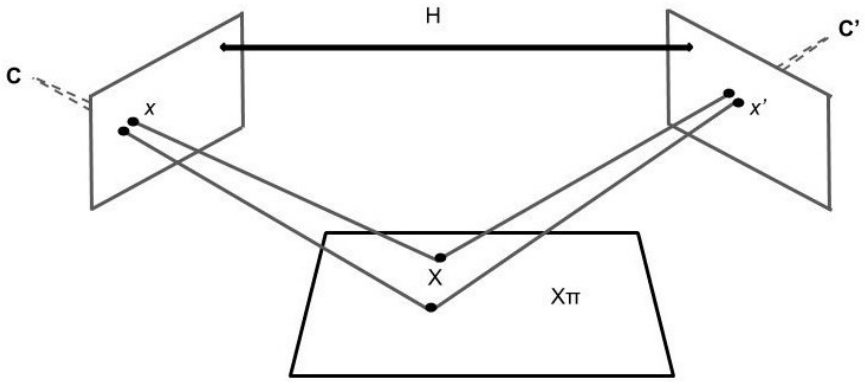


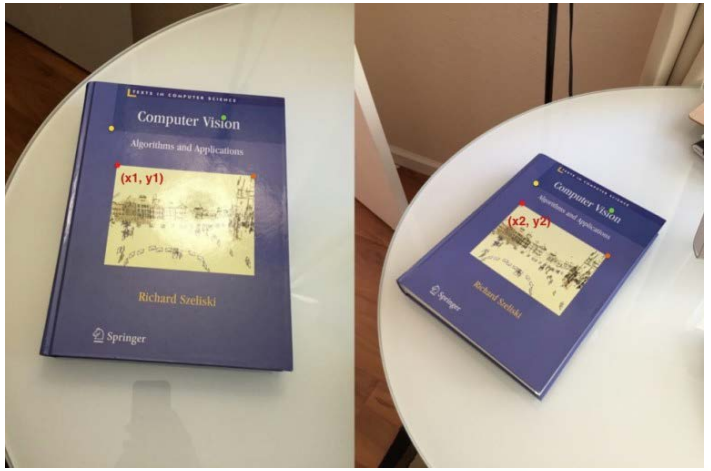


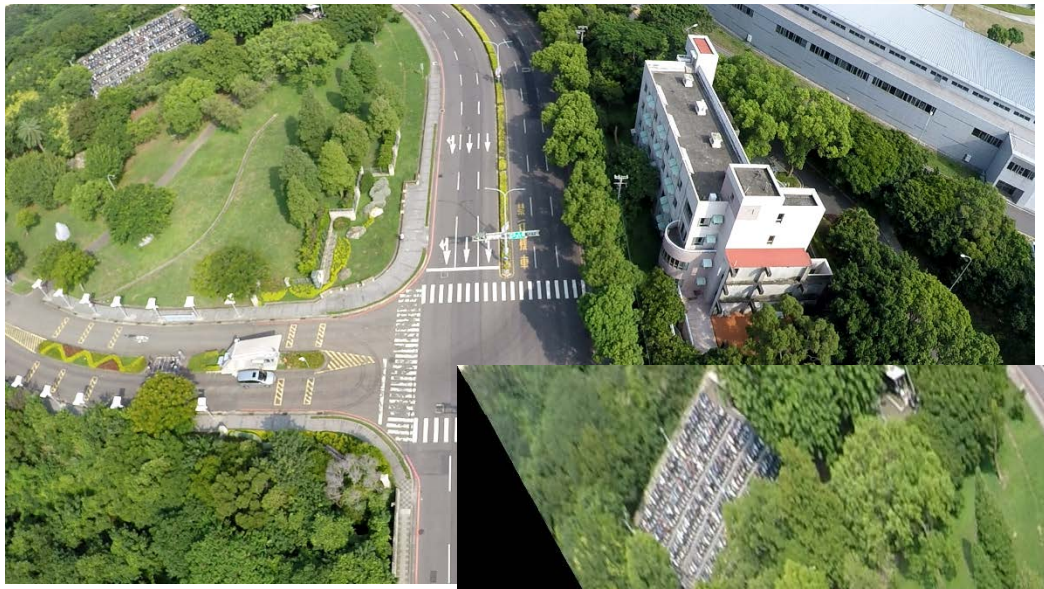
Distortion Removal

Planar Homography

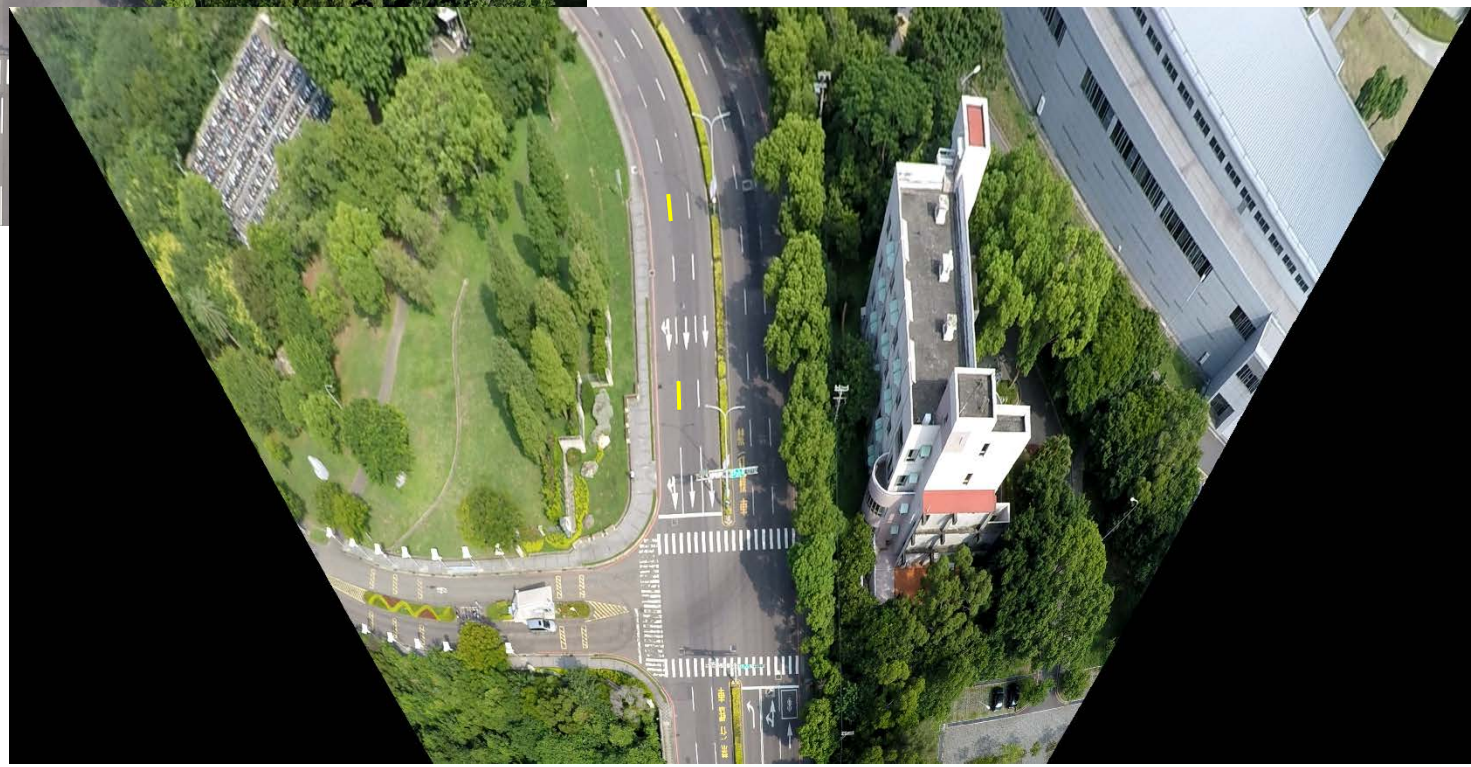
- "homography maps images of points which lie on a world plane from one camera view to another"
  - Siddharth Agarwal, "Homography - And how to calculate it?" All Things about Robotics and Computer Vision, Medium.
- An example of virtual logo applications
  - https://thumbs.gfycat.com/JovialTornIcterinewarbler-mobile.mp4







Perspective Projection by Homography Mapping



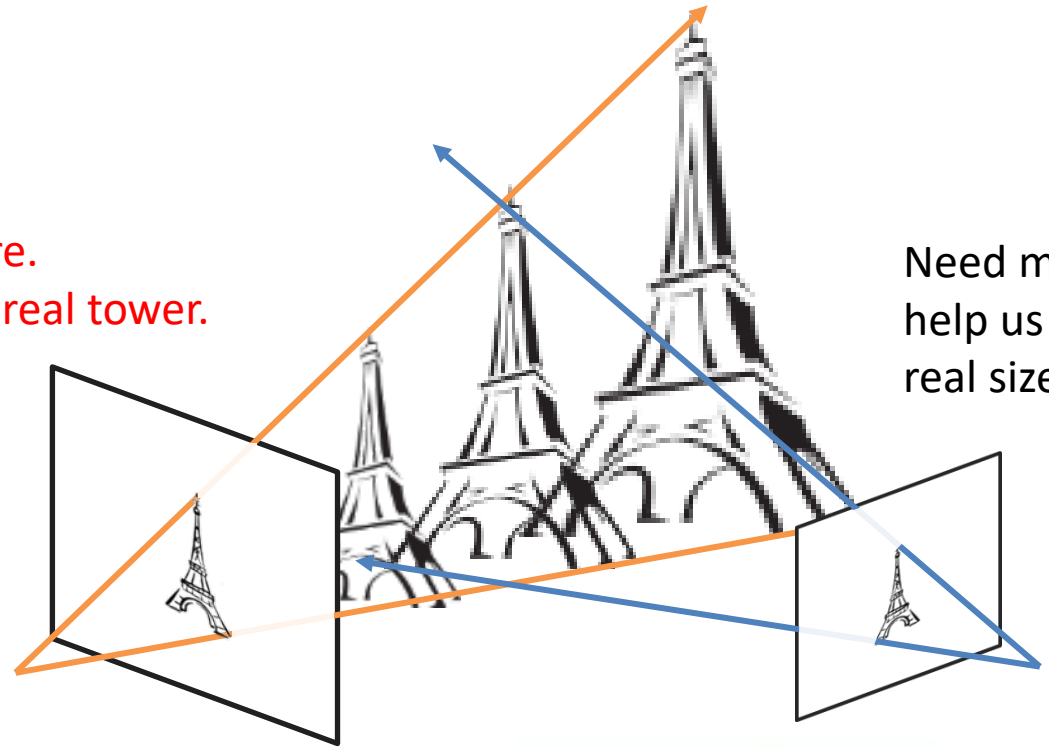
Top-view

3D Positioning by Cameras

- An object's position can be calculated from images taken from different locations.

Which one?

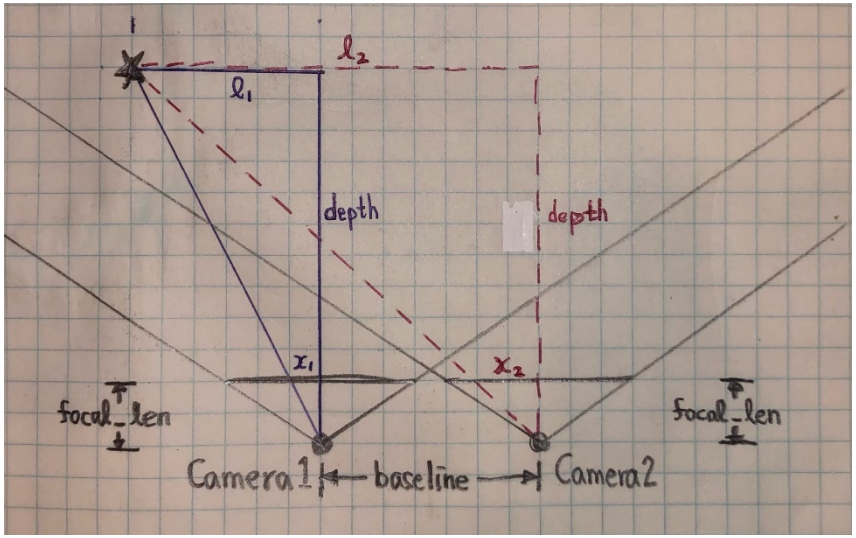
It can be a figure.
It also can be a real tower.



Need more information help us to determine the real size of the object.

Stereo Camera

- Emulate the human binocular vision to provide 3D positioning



$$l_2 - l_1 = baseline; x_2 - x_1 = disparity$$

$$\frac{x_1}{focal\_len} = \frac{l_1}{depth}$$

$$\frac{x_2}{focal\_len} = \frac{l_2}{depth}$$

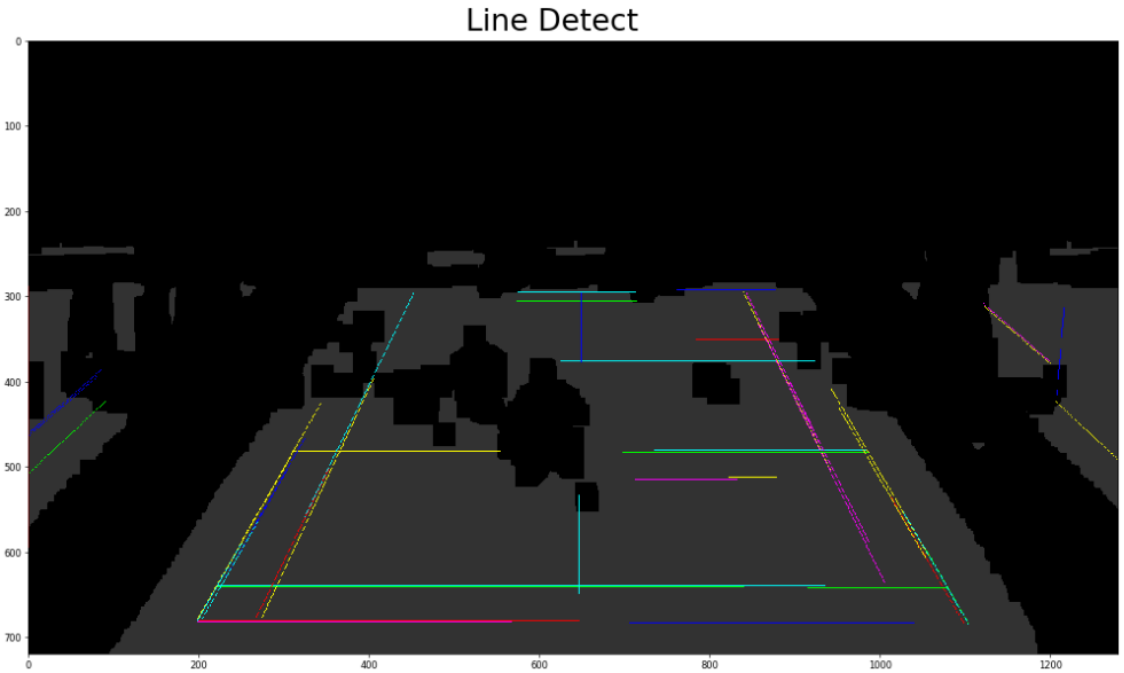
$$\frac{x_2 - x_1}{focal\_len} = \frac{baseline}{depth}$$

$$depth = \frac{baseline \times focal\_len}{disparity}$$

This section will focus on object detection and tracking, and will cover traditional geometric objects detection, deep learning based object recognition and positioning [8,9] as well as object tracking [10,11], and event detection in the temporal dimension [14,16]. The content may include line detection, ellipse detection, deep learning networks for object detection, object tracking algorithms, small object detection and tracking, 2D/3D skeleton detection [12,13], etc. These techniques are commonly used for analyzing sports images [14,15].

PART 3: OBJECT DETECTION AND TRACKING

Geometric Object Detection



Canny Edge Detector

Line Detection (Hough)

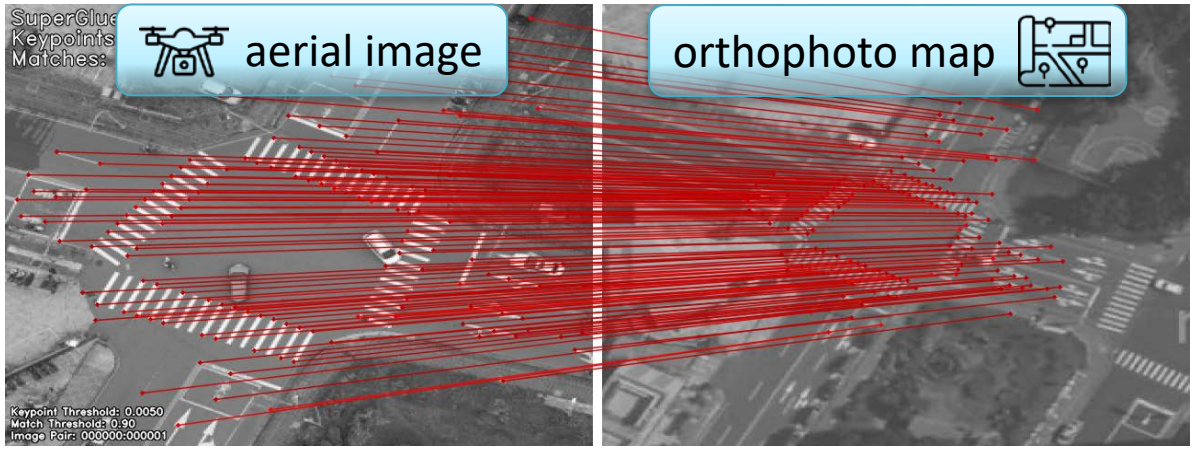
Circle/Ellipses Detection

reference

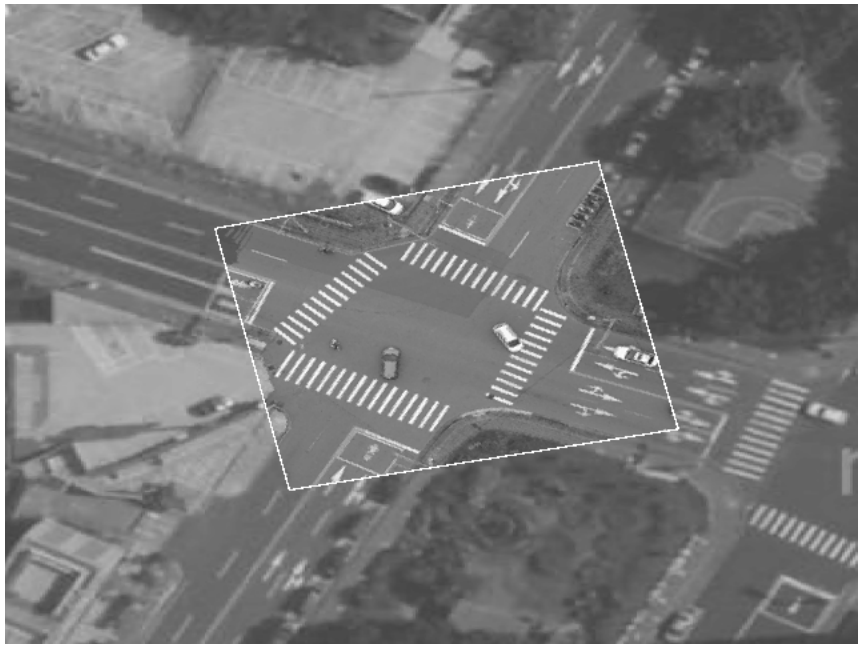
https://en.wikipedia.org/wiki/Hough_transform

Q. Chen, H. Wu, and T. Wada, “Camera Calibration with Two Arbitrary Coplanar Circles”

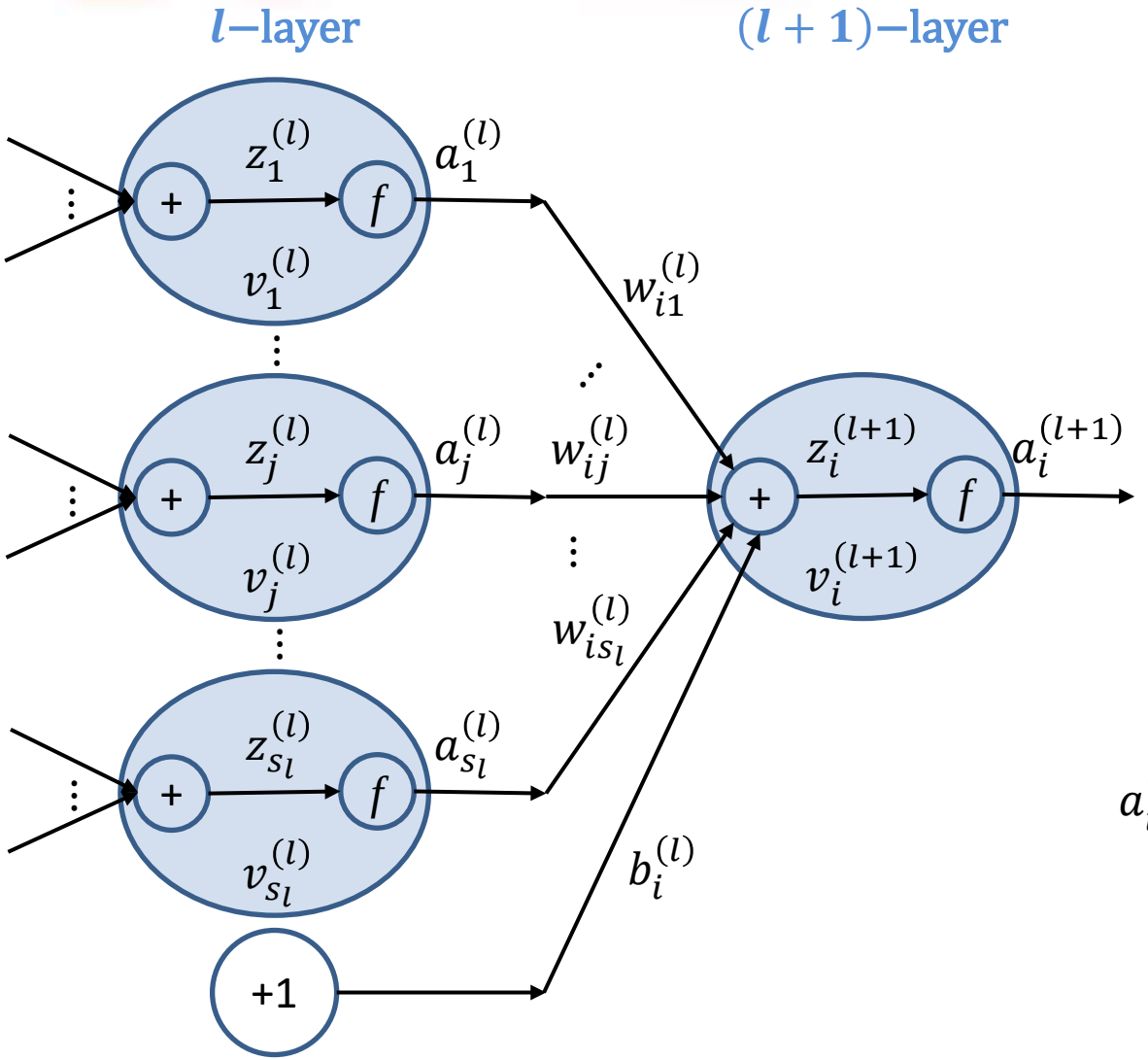
Keypoints



1. Keypoint detection (SuperPoint)
2. Keypoint matching (SuperGlue)
3. Images overlapping or stitching (homography mapping)



Neuron



Weighted sum (linear)

$$z_i^{(l+1)} = \left(\sum_{j=1}^{s_l} w_{ij}^{(l)} a_j^{(l)} \right) + b_i^{(l)}$$

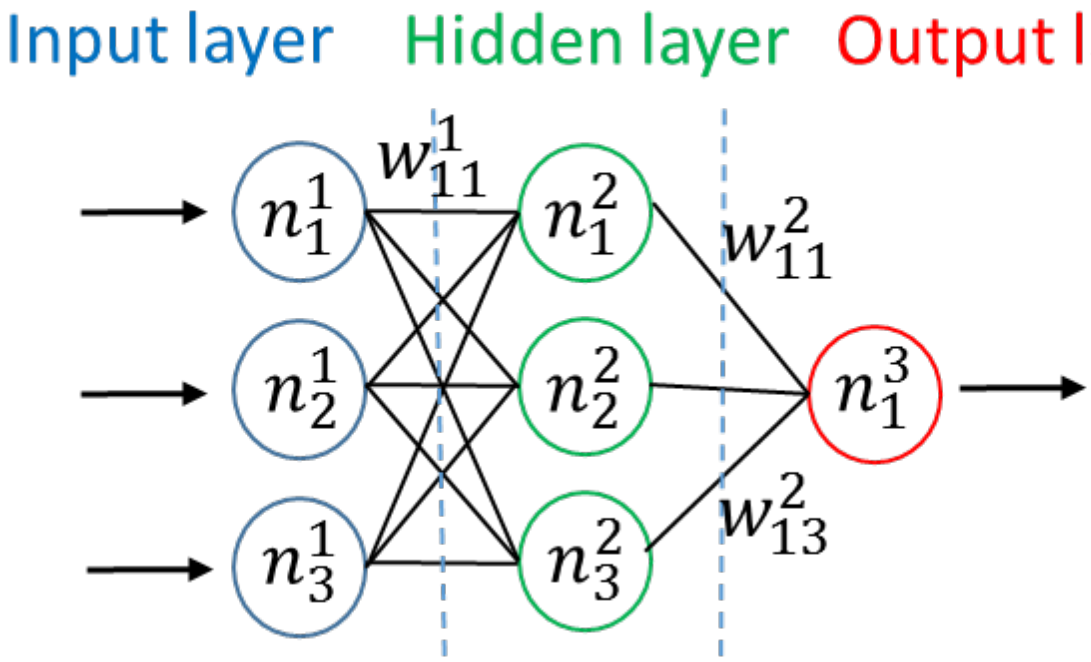
Activation function (non-linear)

$$a_i^{(l+1)} = f(z_i^{(l+1)})$$

Neuron

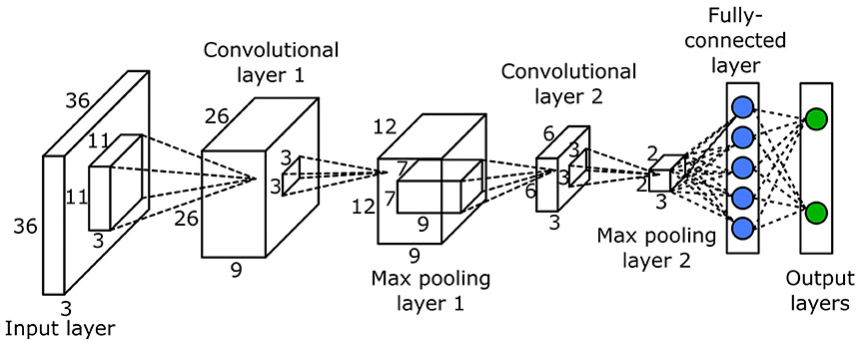
$$a_i^{(l+1)} = f\left(\left(\sum_{j=1}^{s_l} w_{ij}^{(l)} a_j^{(l)} \right) + b_i^{(l)} \right)$$

An Example of 3-layers ANN



Object Classification and Detection

- 2012 AlexNet
  - winner of the 2012 ILSVRC (ImageNet Large-Scale Visual Recognition Challenge)
  - top 5 test error rate 15.4% (the next best 26.2%)
  - 1998 LeCun's work on document recognition

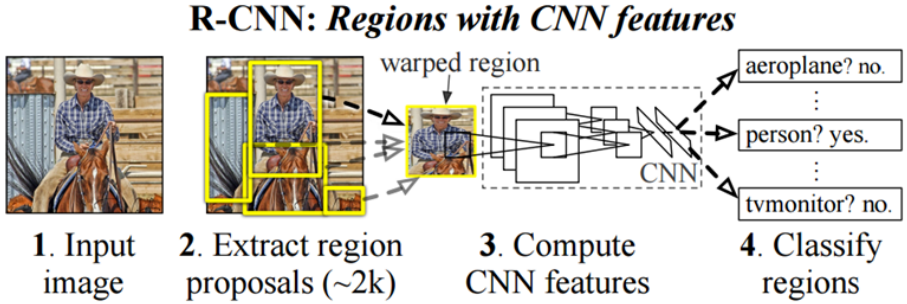


Convolutional Neural Networks

- 2013 ZF Net
  - winner of the 2013 ILSVRC with error rate 11.2%
  - DeConvNet: provide visualization approach to explain the inner workings of CNNs, and insight for improvements to network architectures
- 2014 VGG Net
  - 19 layer CNN achieving error rate 7.3% but not the winners of ILSVRC 2014
  - keep the network deep and simple
- 2014 GoogLeNet
  - 22 layer CNN with a top 5 error rate of 6.7%, winner of ILSVRC 2014
  - not everything is happening sequentially

Source: http://www.kdnuggets.com/2016/09/9-key-deep-learning-papers-explained.html

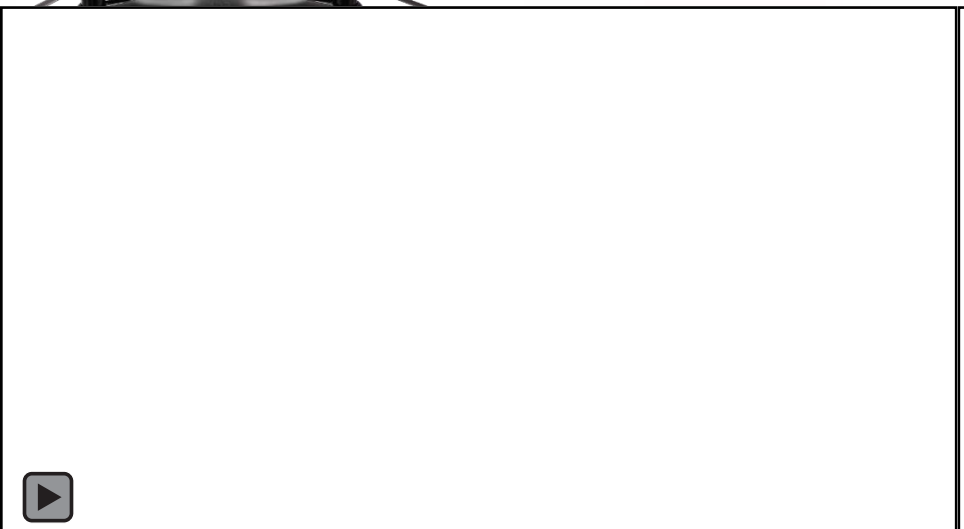
- 2015 Microsoft ResNet
  - 152 layer network architecture that won ILSVRC 2015 with an error rate of 3.6% (humans generally hover around a 5-10% error rate)
  - introduce the idea of residual learning
- Region-based CNNs, including R-CNN (2013), Fast R-CNN (2015), and Faster R-CNN (2015)

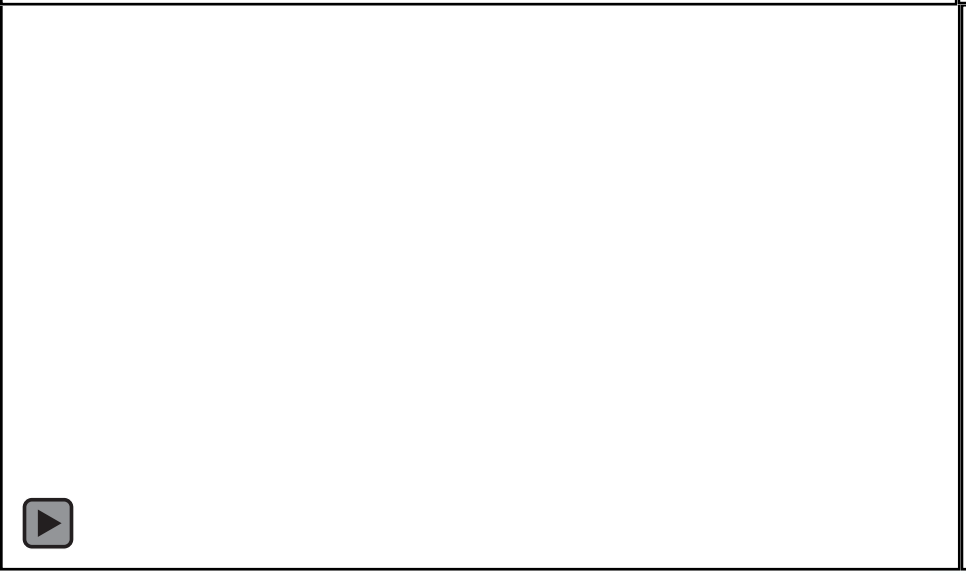


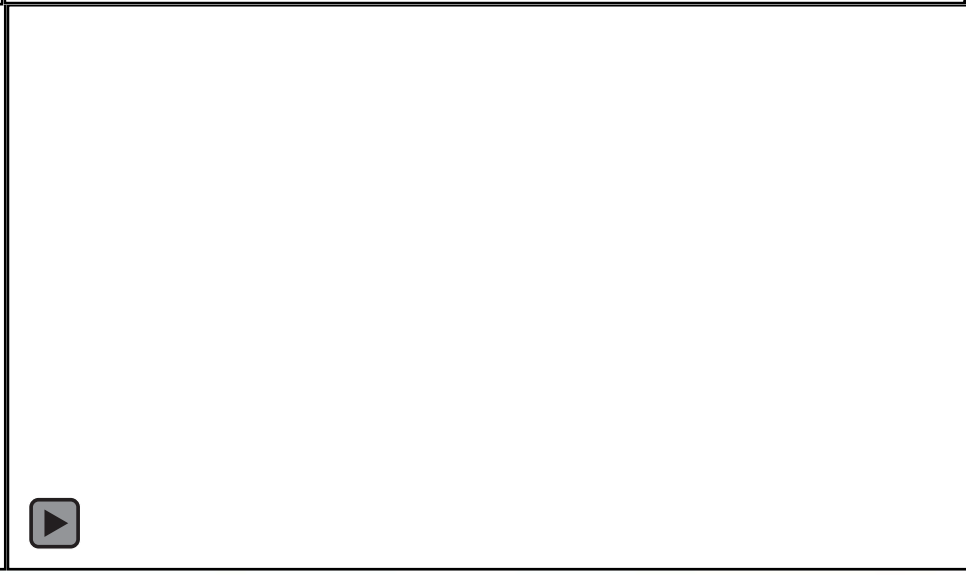
R-CNN workflow

  - faster R-CNN has become the standard for object detection programs today.
- 2014 GAN (Generative Adversarial Networks)
  - adversarial examples are basically the images that fool ConvNets
  - the generator is trying to fool the discriminator while the discriminator is trying to not get fooled by the generator
- 2015 YOLO (You Only Look Once)
  - Grid-based end-to-end object detection CNN

Object Tracking





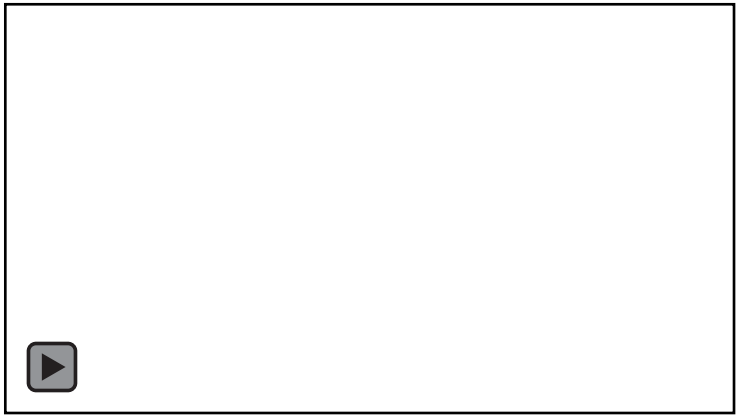


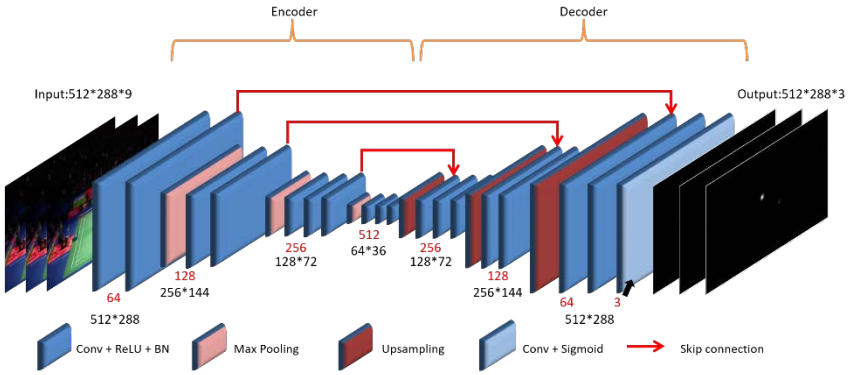
Ball Tracking

Large objects (YOLOv5)

https://medium.com/@gonzacor/ball-tracking-and-detection-in-sports-with-new-yolov5-9f30f5252cf2

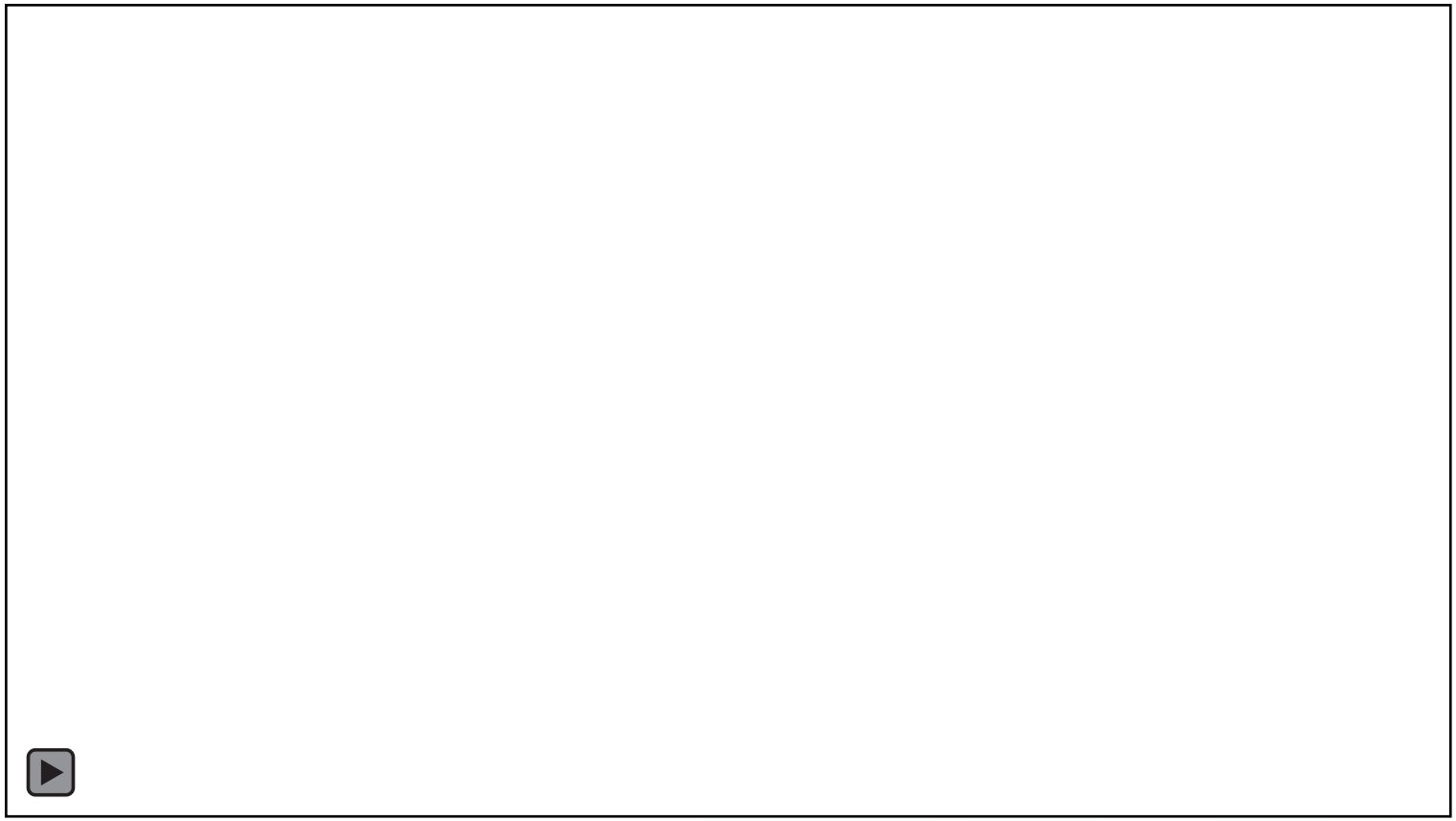
Tiny and speedy object





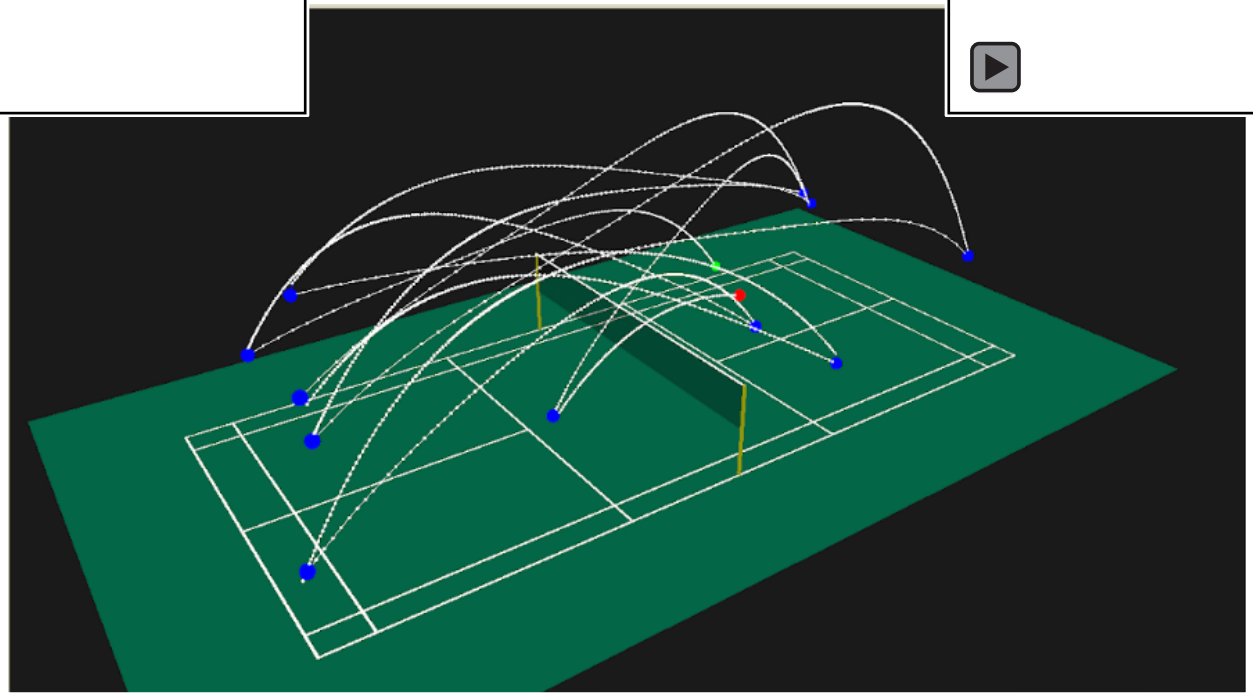
Shot Event Detection

serve hit other ground

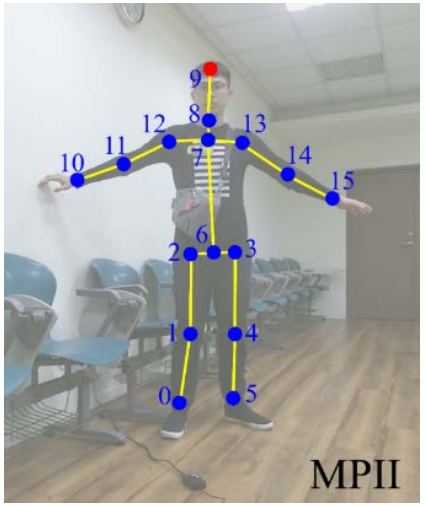


3D Tracking System

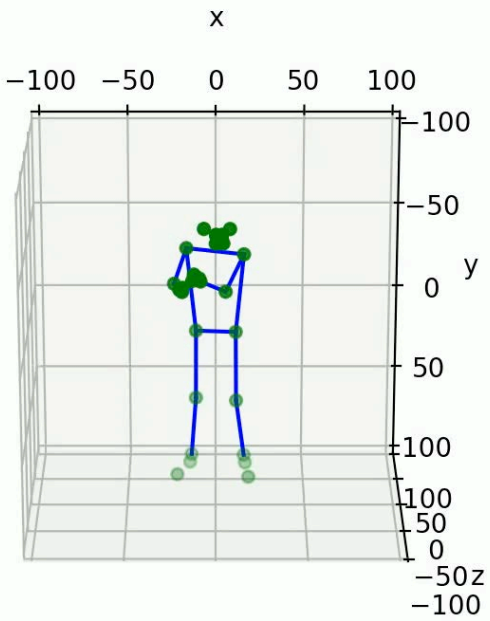
3D Shuttlecock Trajectory
and
Shot Event Detection



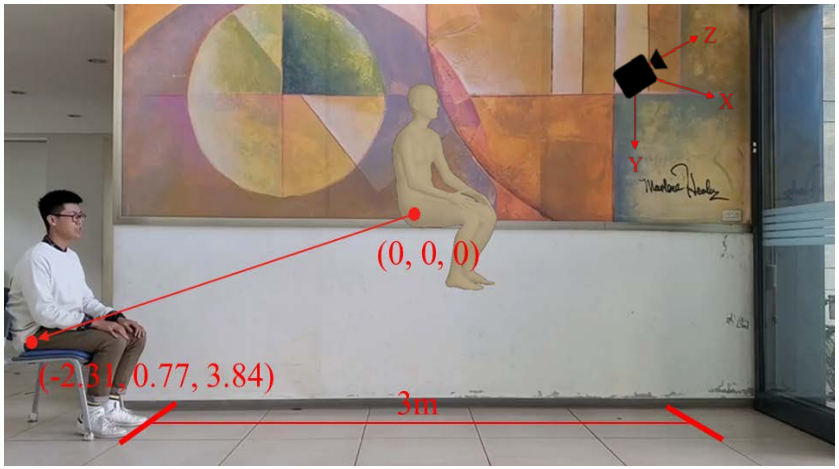
Skeleton Detection



2D Skeleton



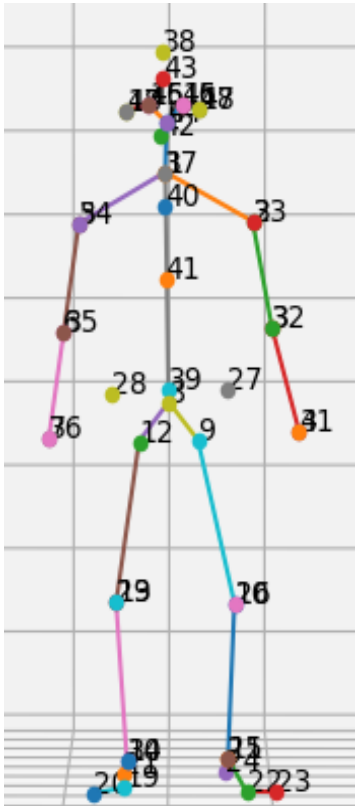
3D Skeleton



3D Rendering

Dataset: MPII, NYCU, …
Networks: HumanPose, OpenPose, VIBE, MediaPipe, …

Skeleton Model (VIBE)

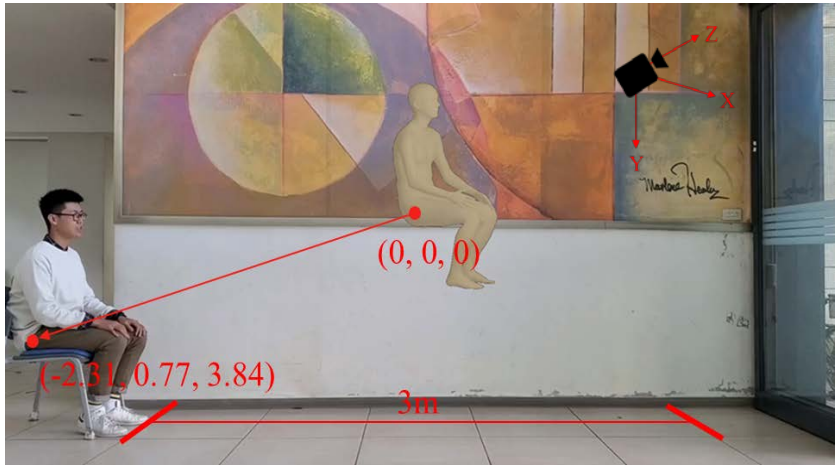


- OpenPose
- H36M
- MPII
- COCO

Joint	Index	Joint	Index	Joint	Index
Nose	#0 #44	Neck	#1 #37	RShoulder	#2 #33
Relbow	#3 #32	RWrist	#4 #31	LShoulder	#5 #34
LElbow	#6 #35	LWrist	#7 #36	MidHip	#8 #39
RHip	#9 #27	RKnee	#10 #26	RAnkle	#11 #25
LHip	#12 #28	LKnee	#13 #29	LAnkle	#14 #30
REye	#15 #46	LEye	#16 #45	REar	#17 #48
LEar	#18 #47	LBigToe	#19	LSmallToe	#20
LHeel	#21	RBigToe	#22	RSmallToe	#23
RHeel	#24	HeadTop	#38	Thorax	#40
Spine	#41	Jaw	#42	Head	#43

3D Coordinate (VIBE)

- The movement of the object can be known by transforming VIBE to camera coordinates.
  - Camera Coordinate
    - Origin: Camera position
    - Metric: Meter

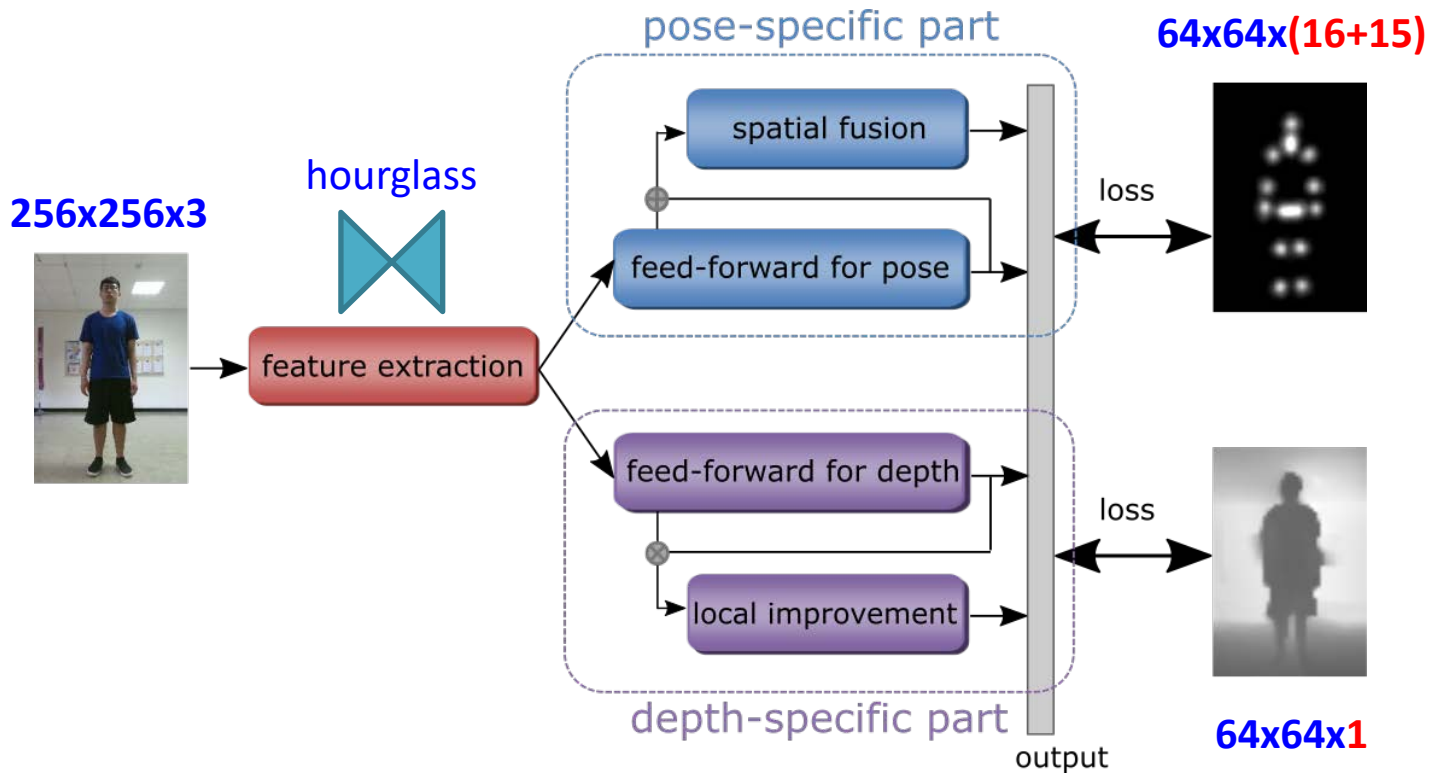


weak perspective camera parameters

Transformation matrix | VIBE Coor. | Camera Coordinate

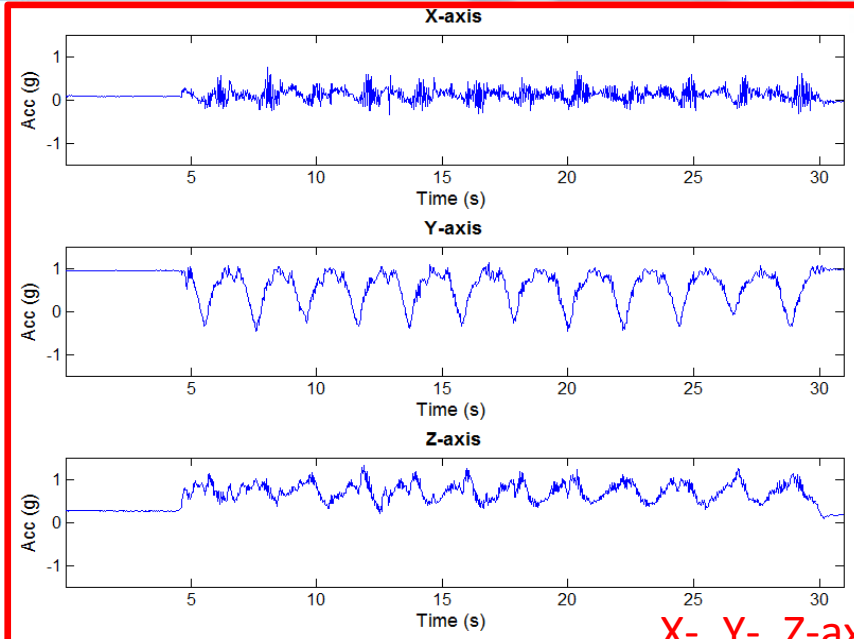
$$\begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ 1 \end{bmatrix} = \begin{bmatrix} a + t_x \\ b + t_y \\ c + t_z \\ 1 \end{bmatrix}$$

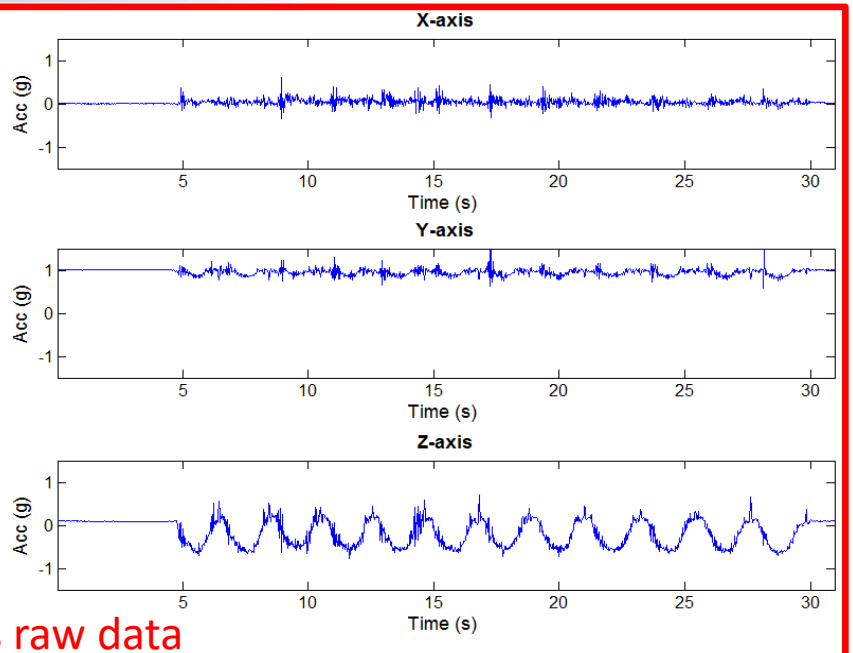
Skeleton Estimation and Depth Prediction (KINET)



Principal Components Analysis (PCA)

Waveforms of the sensor at the right hand wrist

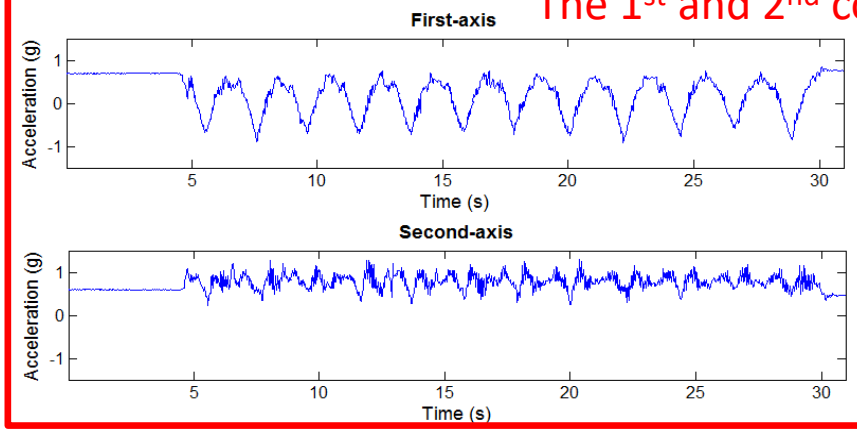


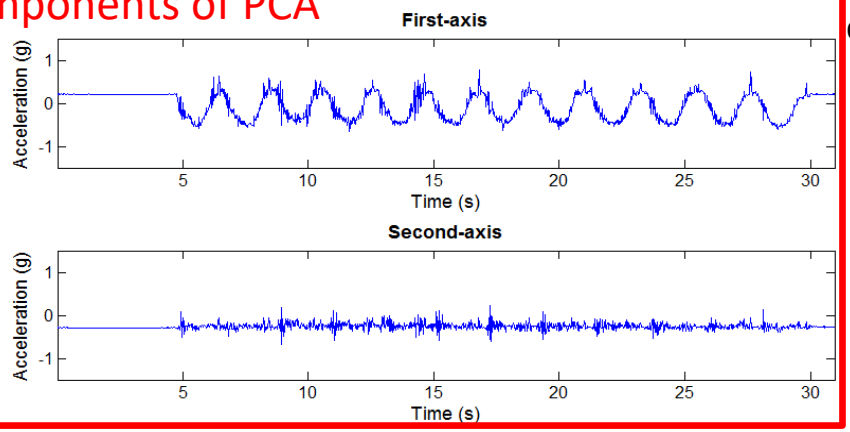


Waveforms of the sensor at the right foot ankle

X-, Y- ,Z-axis raw data

The 1st and 2nd components of PCA

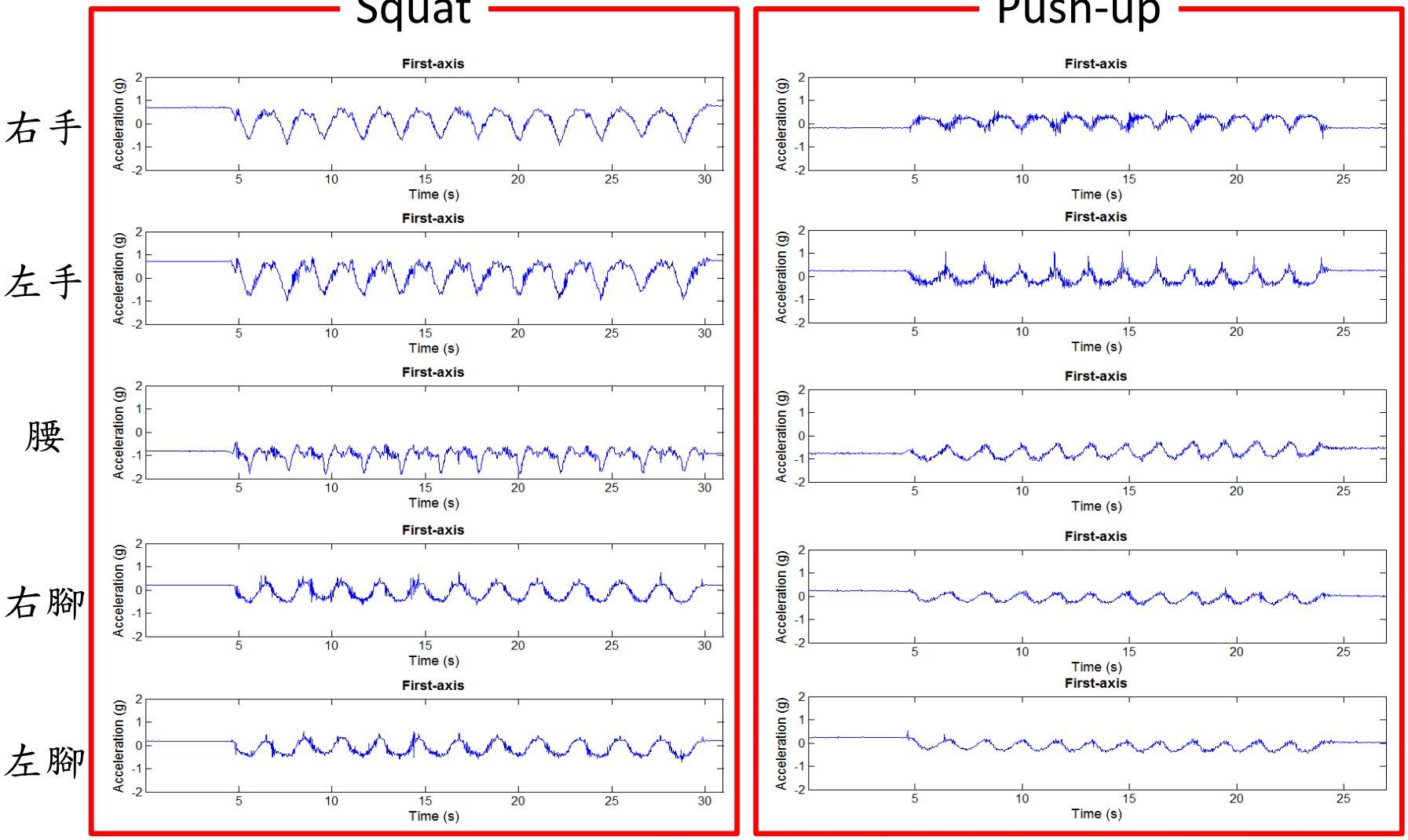




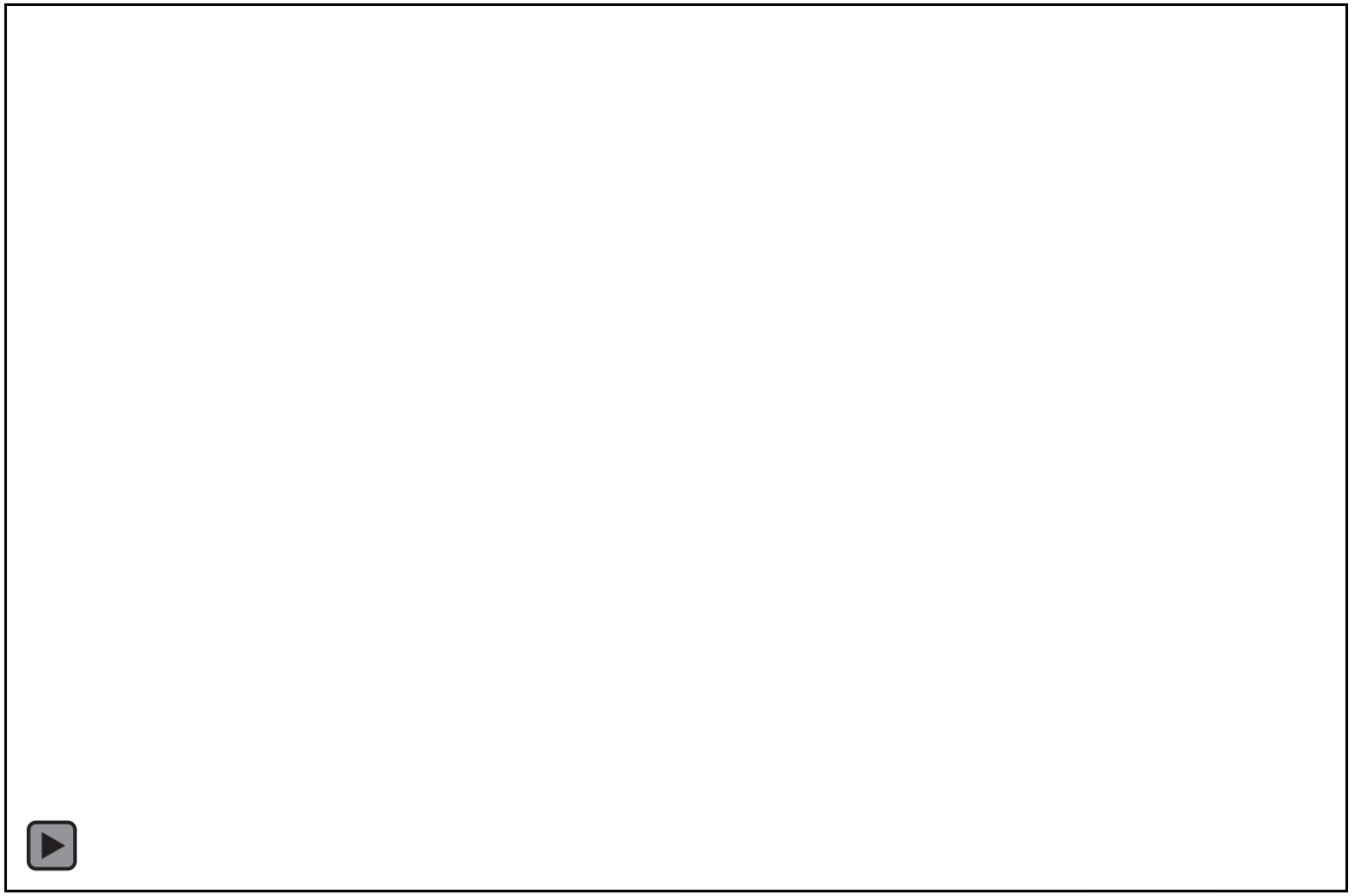
The 1st and 2nd components of PCA (Right Hand)

The 1st and 2nd components of PCA (Right Foot)

Examples of PCA



Action Segmentation and Recognition



To promote the application of these technology and accelerate technology development, in the second half of this tutorial, we will introduce some related practical implementations, which are mostly based on open-source projects. These contents are mostly related to the topics covered in the first three sections.

PART 4: IMPLEMENTATION TUTORIAL (OPEN SOURCE AND OPEN DATA)

Speaker

Yung-Chang Huang
Master of Department of Computer Science
National Yang Ming Chiao Tung University

Email: jason880102@nycu.edu.tw

Introduction

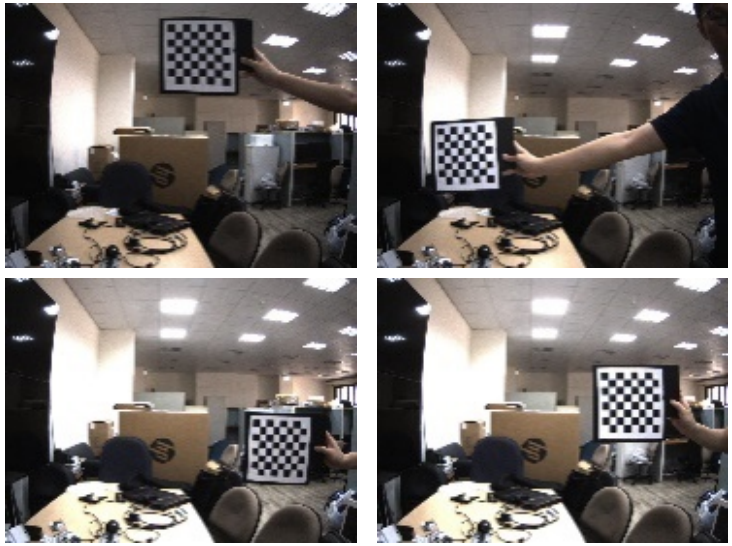
- Camera Geometry
  - Intrinsic & Distortion
  - Extrinsic
  - Homography mapping
- Detection
  - Court Detection
  - Shuttlecock Detection
  - Human Detection
  - Skeleton Estimation
  - Shot Detection

Intrinsic - Resources

- Data
  - https://github.com/Jason-Huang-0102/PAKDD_tutorial/tree/21bddc9c2fdf1b4f358ab0e786d613964603af7d/Intrinsic
- Reference
  - https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html#ga93efa9b0aa890de240ca32b11253dd4a
  - https://towardsdatascience.com/what-are-intrinsic-and-extrinsic-camera-parameters-in-computer-vision-7071b72fb8ec

Chessboard Method

- Input
  - 20 chessboard images with distortion
- Output
  - camera intrinsic matrix
  - Distortion coefficients (used in next chapter)



Chessboard images

$$k = \begin{bmatrix} f & 0 & x \\ 0 & f & y \\ 0 & 0 & 1 \end{bmatrix}$$, f=focal length

, (x,y)=principle points in image

Distortion coefficients=$(k_1\ k_2\ p_1\ p_2\ k_3)$

Implementation

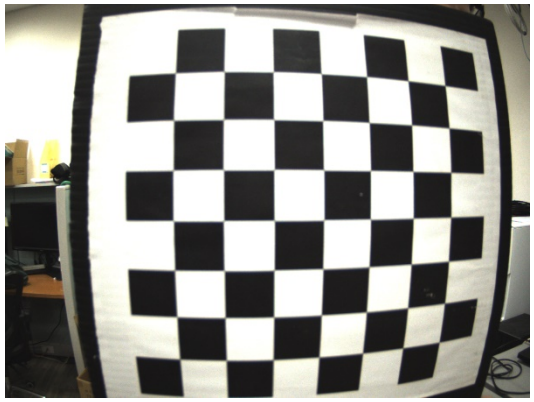
- Import library and setting parameter
- Get the intersection of chessboard
- Calculate camera intrinsic matrix
- Code
  - https://colab.research.google.com/drive/1C8x1CNL-j_bvV5PtTW0pbsqBdacBmhrj?usp=share_link

Distortion - Resources

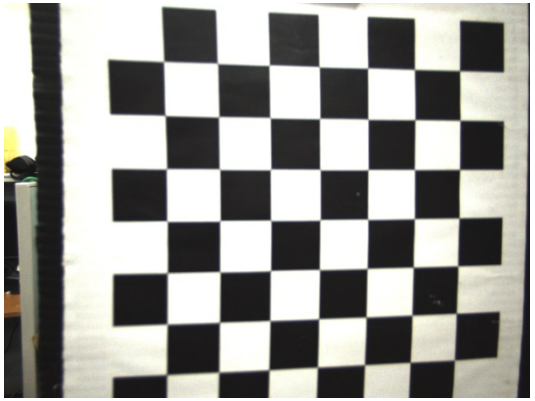
- Data
  - https://github.com/Jason-Huang-0102/PAKDD_tutorial/tree/21bddc9c2fdf1b4f358ab0e786d613964603af7d/Distortion
- Reference
  - https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html
  - https://docs.opencv.org/3.4/d9/d0c/group__calib3d.html#ga7a6c4e032c97f03ba747966e6ad862b1

Undistort Method

- Input:
  - A image with distortion
  - Distorted coefficient
  - Intrinsic matrix
- Output:
  - A undistorted image



Distorted image



Undistorted image

Implementation

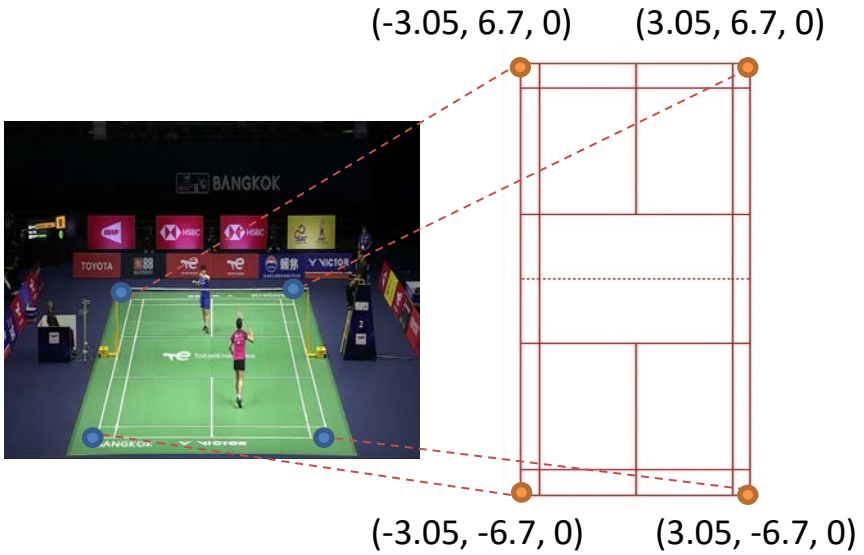
- Import library and setting parameter
- Using OpenCV undistort function undistort the image
- Code
  - https://colab.research.google.com/drive/1BO8r17oXwE_-u-1o_hAIQchecjdfqo55

Extrinsic - Resources

- Data
  - https://github.com/Jason-Huang-0102/PAKDD_tutorial/tree/21bddc9c2fdf1b4f358ab0e786d613964603af7d/Extrinsic
- Reference
  - https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html#ga549c2075fac14829ff4a58bc931c033d
  - https://towardsdatascience.com/what-are-intrinsic-and-extrinsic-camera-parameters-in-computer-vision-7071b72fb8ec

SolvePnP Method

- Input:
  - At least four points in image coordinate
  - At least four points in world coordinate
  - Intrinsic matrix without distortion
- Output:
  - Extrinsic matrix



Extrinsic matrix= $\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}$,

r = rotation matrix,

t = translation matrix

Implementation

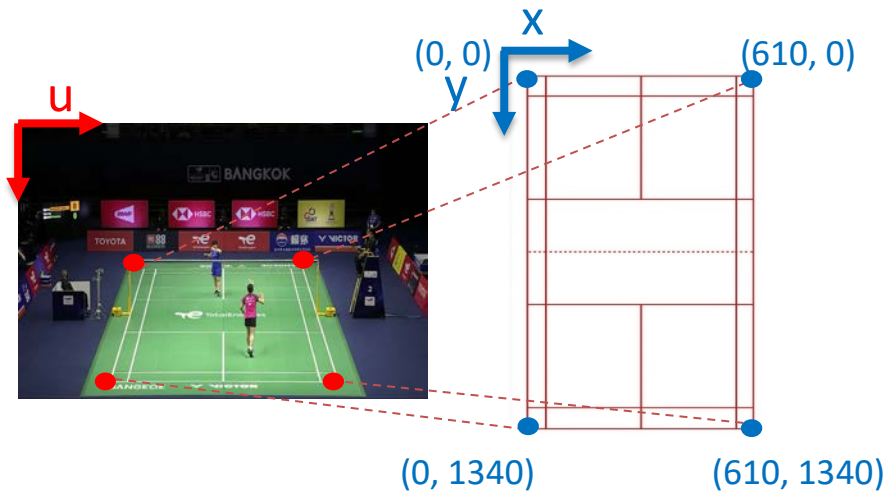
- Import library and setting parameter
- Using SolvePnP calculate camera Extrinsic matrix
- Code
  - https://colab.research.google.com/drive/14nc8uSvBG_3MZNOdhVvFHS3X2v9uuMHU

Homography Mapping - Resources

- Data
  - https://github.com/Jason-Huang-0102/PAKDD_tutorial/tree/21bddc9c2fdf1b4f358ab0e786d613964603af7d/Homography
- Reference
  - https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html#ga549c2075fac14829ff4a58bc931c033d
  - https://docs.opencv.org/4.x/d5/d1f/calib3d_solvePnP.html

Least Square Method

- Input:
  - At least four points in pixel coordinate
  - At least four points in court coordinate
- Output:
  - Homography matrix



$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

$$x = \frac{x'}{w'} = \frac{a_{11}u + a_{12}v + a_{13}w}{a_{31}u + a_{32}v + a_{33}w}$$

$$y = \frac{y'}{w'} = \frac{a_{21}u + a_{22}v + a_{23}w}{a_{31}u + a_{32}v + a_{33}w}$$

Implementation

- Import library and select points
- Using FindHomogaphy calculate camera Homography matrix
- Code
  - https://colab.research.google.com/drive/1HgT7QjaDo9PEVZbMXFn6cV_Lu5QaFSVp

Court Detection - Resources

- Data
  - https://github.com/Jason-Huang-0102/PAKDD_tutorial/tree/21bddc9c2fdf1b4f358ab0e786d613964603af7d/CourtDetection
- Reference
  - https://docs.opencv.org/3.4/da/d22/tutorial_py_canny.html
  - https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html

Canny + Hough

- Input:
  - A image
- Output:
  - A image with court layout

Implementation

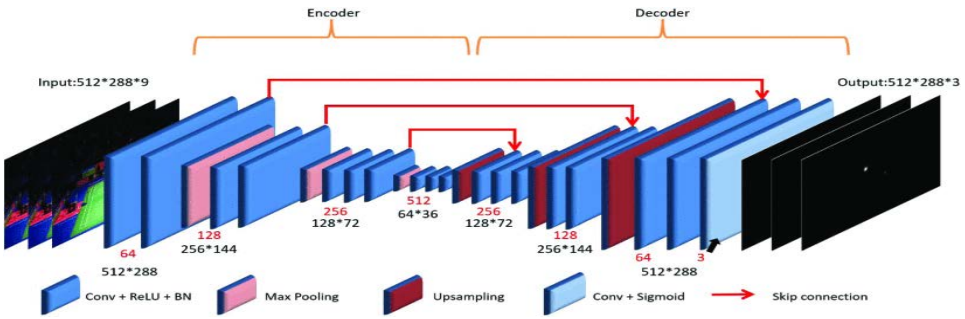
- Import library
- Image preprocessing
- Find lines and the intersection points
- find the court corner
- Code
  - https://colab.research.google.com/drive/1WGLoqgrpySeTDKdjF4hhOjQZDUM1XOof

Shuttlecock Detection - Resources

- Data
  - https://github.com/Jason-Huang-0102/PAKDD_tutorial/tree/21bddc9c2fdf1b4f358ab0e786d613964603af7d/ShuttlecockDetection
- Reference
  - https://nol.cs.nctu.edu.tw:234/open-source/TrackNetv2
  - https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9302757

TrackNetV2

- Input
  - A rally video
- Output
  - A rally video with detected shuttlecock position



Implementation

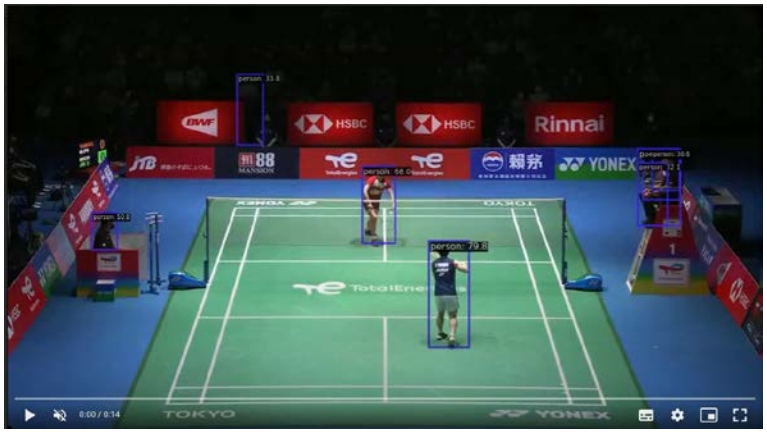
- Import library and model
- Prediction & output video and .csv file
- Code
  - https://colab.research.google.com/drive/1oQBXijUuXibNzVnnX-98pN2DUhed19C3

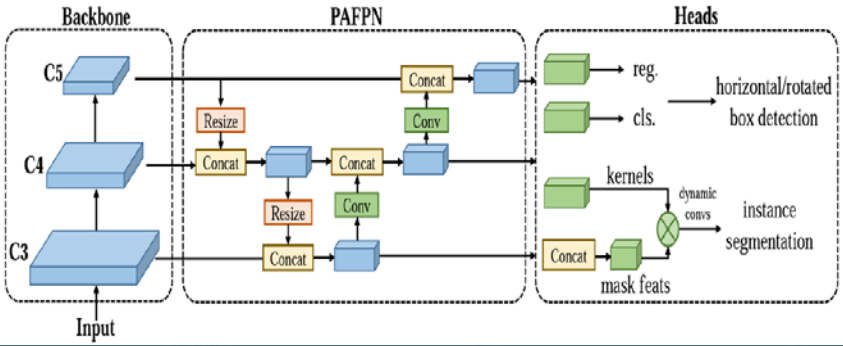
Human Detection - Resources

- Data
  - https://github.com/Jason-Huang-0102/PAKDD_tutorial/tree/21bddc9c2fdf1b4f358ab0e786d613964603af7d/HumanDetection
- Reference:
  - https://github.com/open-mmlab/mmdetection
  - https://mmdetection.readthedocs.io/en/latest/
  - https://arxiv.org/abs/1906.07155

RTMDet

- Input:
  - A rally video in games
- Output:
  - A rally video with human bounding box





Implementation

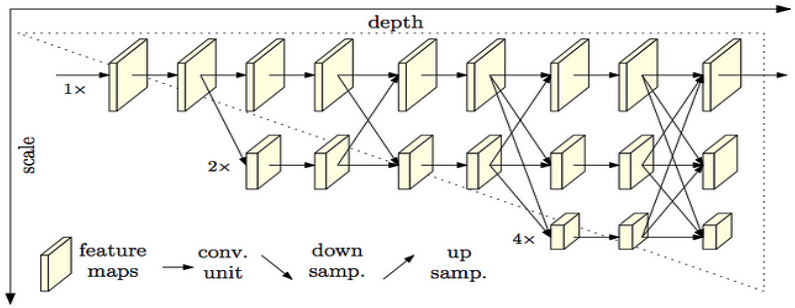
- Build Environment and install MMDetection
- Data Preprocess
- Predict
- Code
  - https://colab.research.google.com/drive/1AqvS8LOrjyz3tP7p5--hUUrVsTn4W6Iw

Skeleton Estimation - Resources

- Data
  - https://github.com/Jason-Huang-0102/PAKDD_tutorial/tree/main/SkeletonEstimation
- Reference:
  - https://github.com/open-mmlab/mmpose
  - https://mmpose.readthedocs.io/en/latest/
  - https://towardsdatascience.com/hrnet-explained-human-pose-estimation-sematic-segmentation-and-object-detection-63f1ce79ef82

HRNet

- Input:
  - A rally video
- Output:
  - A rally video with skeleton



Implementation

- Build Environment and install MMPose
- Get people using MMDetection
- Get skeleton using MMPose
- Code(MMPose official)
  - https://colab.research.google.com/github/open-mmlab/mmpose/blob/master/demo/MMPose_Tutorial.ipynb

Shot Detection - Resources

- Data
  - https://github.com/Jason-Huang-0102/PAKDD_tutorial/tree/21bddc9c2fdf1b4f358ab0e786d613964603af7d/ShotDetection
- Reference
  - https://github.com/kwyoke/Badminton-hit-detection
  - https://arxiv.org/abs/2204.01899

HitNet

- Input:
  - Four corner of court
  - Shuttlecock position
  - Skeleton of players
- Output:
  - A video with shot detection result

Implementation

- Data preprocessing
- Define hyperparameters
- Construct the sequence model
- Inference
- Code
  - https://colab.research.google.com/drive/1TZVlmulUdZzY4M_XigxCemzkwoDxNq5-

Thanks for Listening