Condor: Mobile Golf Swing Tracking via Sensor Fusion using Conditional Generative Adversarial Networks

Hong Jia[†‡], Jun Liu[†], Yuezhong Wu[†‡], Tomasz Bednarz[†], Lina Yao[†], and Wen Hu[†]

[†]University of New South Wales

[‡]Data61 CSIRO

{h.jia, t.bednarz, lina.yao, wen.hu}@unsw.edu.au, {jun.liu, yuezhong.wu}@student.unsw.edu.au

Abstract

This paper explores the possibility of incorporating sensor-rich and ubiquitously deployed mobile devices into sports analytics, particularly to the game of golf. We develop a novel solution to track a player's swing in three-dimensional (3D) space using inexpensive tools such as depth sensors and Inertial Measurement Units (IMUs). Existing solutions based on these devices cannot produce consistent and accurate swing-tracking. This is due to commonly known issues with occlusion and low sampling rates generated by depth sensors and complex IMU noise models.

To overcome these limitations, we introduce *Condor*, a tailored deep neural networks to make use of sensor fusion to combine the advantages of these two sensor modalities, where IMUs are not affected by occlusion and can support high sampling rates and depth sensors produce more accurate motion measurements than those produced by IMU. *Condor* could be implemented with edge devices such as a smart wristband and a smartphone, which are ubiquitously available, for accurate golf swing analytics (e.g. tracking, analysis and assessment) in the wild. Our comprehensive experiment shows that proposed method outperforms other solutions and reaches 6.57 cm error in subject-dependent model and less than 10 cm error for unknow-subjects via a tailored conditional Generative Adversarial Networks (cGAN).

Categories and Subject Descriptors

C.5 [**COMPUTER SYSTEM IMPLEMENTATION**]: Miscellaneous; I.5 [**PATTERN RECOGNITION**]: Models

General Terms

Design, Models

Keywords

Mobile Computing, Golf Swing Tracking, Ubiquitous

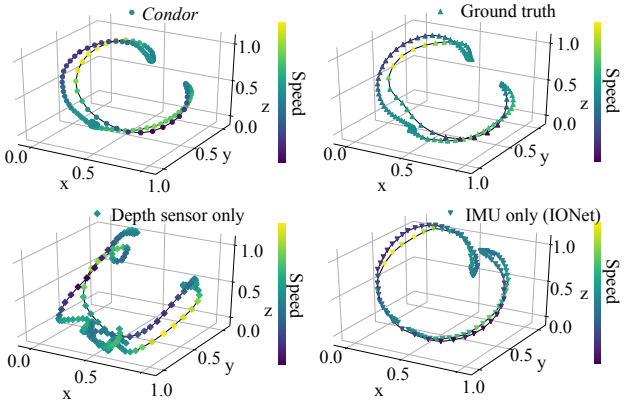

Figure 1. The 3D trajectory comparison of a golf swing for *Condor*, ground truth, depth sensor and IMU only. Obviously, *Condor* shares the most correct details during all time stamps with ground truth. Depth sensor, however, cannot tell a smooth tracking due to occlusion. Only using IMU (the state-of-the-art) bears large offset error due to its natural mechanism.

1 Introduction

Sports analytics is a thriving industry with its market size estimated to reach five billion dollars within the next two years [14]. Swing tracking is one important aspect that provides the sector with key information in many sports assessments. However, given that current vision-based approaches typically suffer from low sampling rates, occlusion, and intensive computational resource requirements (such as heavy GPU workloads) [29], mobile and inexpensive sports analytics pose a very challenging task. For example, state-of-the-art motion capture systems such as OpenPose [6] uses five video cameras to avoid the occlusion [27] and reaches an 3 cm mean average error (MAE). Obviously, multi-camera is inconvenient for mobile golf swing tracking. Meanwhile, singular video-based solutions usually miss key information during the swing-phase transition due to occlusion (i.e., the point of interest being tracked is blocked by other body parts from the camera's view), which reduces their accuracy significantly [30, 38]. In fact, professional players in sports such as golf can produce swings up to 67 m/s, which makes

International Conference on Embedded Wireless Systems and Networks (EWSN) 2021
17–19 February, Delft, The Netherlands
ISBN: 978-0-9949886-5-2

singular video-based solution for fast 3D tracking very challenging. In comparison, we propose to using a singular depth sensor and an IMU which are mobile and ubiquitous for swing tracking in golf.

Commercial swing tracking usually relies on marker-based and fixed camera tracking systems such as Hawkeye and Vicon [9]. Based on the markers involved, these multiple, high-speed camera-based systems can produce accurate and fast pose tracking in 3D space. However, these systems cannot operate without markers and are usually very expensive. To this end, researchers are investigating alternative methods such as by using single or multiple Inertial Measurement Units (IMUs) to calculate the location of the point being tracking [11]. Specifically, an IMU captures the acceleration, gyroscope, and compass measurements, after which the trajectory of a point can be calculated through double integration with quaternion calibration — that is, from acceleration to velocity, and then its integral to the position. Unfortunately, while the acceleration sensors in inexpensive IMUs may operate in high sampling rates, they usually suffer from issues with complex noise, which also produces errors [25]. These faults can be significant — especially those accumulated over time — even with carefully calibrated IMUs. Meanwhile, compasses only operate in low sampling rates.

Depth cameras and/or sensors such as Microsoft Kinect can track poses in 3D space. However, current applications and research on depth cameras show that they too suffer from accuracy misalignment, occlusion, and problems with missing frames [18]. Further, existing depth cameras cannot be used for fast swing tracking directly due to their low sampling rates and occlusion [31].

As such, this paper attempts to address the occlusion and low sampling rate problem in current golf swing-tracking approaches using depth sensors. By tailoring a lightweight deep-learning network, the proposed approach (called *Condor*[1]), fuses IMUs and depth sensors to calibrate each other for swing tracking in golf. The *Condor* results, compared with previous works, are shown in Figure 1. During the training stage, *Condor* learns to calibrate IMU measurements with those of the depth sensor as well as the ground truth. During testing, *Condor* reconstructs and predicts the trajectory based on the IMU measurements when the depth sensor is occluded or when it under-samples due to sampling rate limitation. Compared to state-of-the-art commercial motion-tracking systems such as Vicon [24], *Condor* is comparatively cheap to run for swing tracking as it replies on equally inexpensive IMUs and depth sensors built into mobile devices such as Apple Watches, Lenovo Phab 2 Pro Tango-enabled smartphones, and Intel RealSense depth cameras. These mobile devices also enable *Condor* to achieve swing tracking in an outdoor setting (e.g. in the golf course).

Overall, this paper makes the following contributions:

- An architecture known as *Condor* was designed to exploit IMUs for their high sampling rates and non-line-of-sight nature (Section 2). This allowed us to address both the positive and negative aspects of depth sensors (including the occlusion and low sampling rate problem found in most systems), while simultaneously harnessing their advantages to tackle the time-cumulative errors and complex noise model problem found in inexpensive IMUs. With *Condor*, these two modalities actually complement one another during swing tracking in golf. The design features a novel feature extractor that combines information from spatial, temporal, and modality domains (Section 3.1).
- To the best of our knowledge, *Condor* is the very first deep learning framework specifically designed for swing tracking in golf. During a swing, the designed neural network model (Section 3) learns latent relationships between the IMU and depth camera and uses multidimensional sensor measurements to predict swing trajectory at different stages once the motion reaches high speed and the key joints of the skeleton of the player become overlapped. We also introduced a conditional Generative Adversarial Networks (cGAN) [13] model to extract subject independent features and increase user-friendliness of the system (Section 3.3).
- We visualize the importance of different sensor modalities in different stages proves that proposed framework interchangeably utilize different sensor in different stages to realize 3D swing tracking (Section 4.3). We implement *Condor* in a prototype consisting of an off-the-shelf IMU device and a depth sensor. Comprehensive experiments in golf-swing tracking show that the approach reduces errors by more than 60% compared to the state-of-the-art approach. We tested proposed framework is lightweight enough and can be implemented on edge devices (Section 4.7).

2 *Condor* Overview

2.1 Use Cases

Figure 2 shows a typical golf swing, which can be categorized into five stages: setup, top of the swing, impact, follow-through, and finish. Each respective stage is important because it defines a plane of the swing trajectory, including speed, direction, and angle information, all of which direct the ball into different directions and locations.

Importantly, high-speed tracking cameras were set up in many driving ranges and golf club studios to assist players in improving their swing skills. *Condor* makes use of these facilities to calibrate mobile depth sensors and IMUs through a novel deep-learning framework, enabling high-precision swing tracking in practice (i.e., on a golf course).

2.2 System overview

Figure 3 gives an overview of *condor* architecture. First, it collects IMU (Acc_x, Acc_y, Acc_z, Gyr_x, Gyr_y and Gyr_z) and depth (Kw_x, Kw_y and Kw_z) measurements from mobile devices such as smart wristbands and depth cameras before pre-processing them (see Section 2.3). Note that *Condor* does not rely on IMU compass measurements; thus, the input can be denoted as 9-dimension data (Kw_x, Kw_y and Kw_z, Acc_y, Acc_z, Gyr_x, Gyr_y and Gyr_z). The output of the pre-processing module is the concatenation of IMU and depth camera mea-

[1] In the sport of golf, *Condor* is an unofficial name for a score of four under par (-4), and is the lowest individual hole score ever made.

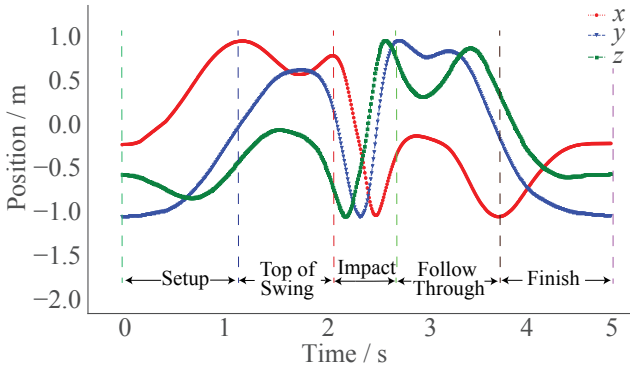

Figure 2. One golf swing in three dimensions (3D). The swing speed reaches 150 mph (67 m/s) and 100 mph (44.7 m/s) for professional golfers and beginners, respectively.

surements (X_0, X_1,..., X_9), which subsequently function as input for the proposed deep-learning network, used to exploit the spatial and temporal structure of different sensor modalities (see Section 3 for details). Finally, the output (or predictions) ($\hat{Pos}_x$, $\hat{Pos}_y$ and $\hat{Pos}_z$) will be highly accurate 3D positions of the tracking point (i.e., the wrist of a player and/or the club) at a given timestamp.

2.3 Signal processing

Pre-processing for *Condor* includes the following components.

2.3.1 Coordinates

Traditional coordinates calculation methods require quaternions for conversion from local coordinates (of the IMU and depth sensors) to world coordinates. The quaternion can be produced based on gravity and the northerly cardinal direction produced by accelerometers and compasses. In contrast, the proposed system automatically converts IMU coordinates to world coordinates using a deep neural network (discussed in Section 3). Since many edge devices such as IMU-based smartband did not have compass (e.g. Samsung Gear S2, Moto 360, etc.), we hereby designing a framework using accelerator and gyroscope only.

2.3.2 Low-pass filtering

We apply a low-pass filter to smooth any noisy IMU measurements. Since depth sensor measurements are less noisy, we use the raw measurements directly.

2.3.3 Synchronization

We use Coordinated Universal Time (UTC) stamps help to organize the measurements captured by different devices (e.g., depth cameras and IMUs). By tagging time stamps, the collected sensor data can be aligned automatically for further preprocessing. Figure 4 shows synchronized measurements from few golf swings.

2.3.4 Segmentation

Common time-series measurement segmentation methods include window sliding and gradient cutting. *Condor* applies a sliding window to segment the signals from the depth sensor coordinates along the z-axis; this is because the gradient varies more along this direction (i.e., roughly corresponding to the direction of gravity changes) than the other two axes (i.e., x- and y-axis). As such, the proposed segments do not have an overlap since swing motions are separable. Example segmentation results are shown in Figure 4.

2.3.5 Resampling

Since different devices have different sampling rates, we align the measurements from the swing segments following cubic interpolation into a fixed number of time stamps T (e.g., $T = 152$ later used for the evaluation in Section 4). Example segmentation results are shown in Figure 5.

3 *Condor* Deep Learning Framework

The *Condor* network was designed with a feature extractor, a generator for regression and a discriminator to remove the subject-related prior (see Figure 3). For a fixed length swing segment that has T time stamps, the dimensions of the input matrix is 9 (i.e, X_0, X_1,..., X_9, see Section 2.2) and T respectively. For the output trajectory of a point being tracked, the dimensions become 3 (i.e, $\hat{Pos}_x$, $\hat{Pos}_y$ and $\hat{Pos}_z$, see Section 2.2) and T respectively. Intuitively, the input has the temporal (IMU), spatial (depth camera) and modality (sensor fusion) information. The Long Short Term Memory network (LSTM) mainly focuses on temporal feature extraction. However, for spatial information, directly converting input data format to time series before inputting them to LSTM usually learns a poor representation [21]. Meanwhile, for spatial information, Convolutional Neural Network (CNN) has been shown successful as it can extract more informative features [16, 26]. Therefore, we propose to use the CNN as the feature extractor to extract more informative spatial features, before inputting these features into a LSTM to predict 3D trajectories. Our extensive experiment in Section 4.2 also demonstrated the superior performance of the proposed approach.

3.1 Feature Extractor

As discussed in Section 2.2, *Condor* concatenates nine axis signals (Kw_x, Kw_y, Kw_z, Acc_x, Acc_y, Acc_z, Gyr_x, Gyr_y and Gyr_z) and inputs them into the neutral network, the relationship between these signals need to be encoded into a *latent representation* (i.e., feature extraction).

CNN is often used for feature extraction purpose in many images applications [16, 33, 36, 26, 26]. In *Condor*, we designed a CNN block as the feature extractor for swing tracking and formulate it as a *regression* problem. Here, the feature extractor has three separated groups of convolutional networks: spatial domain, temporal domain, and modality domain, all of which maximize the input signal's spatiotemporal, intra-modality, and inter-modality feature representation respectively. This is because the measurements in different sensor modalities (i.e., IMU and depth sensor) can change rapidly during swing tracking due to occlusion and/or sensor noise (see the lower two plots in Figure 1). These factors must be considered within the feature extractor architecture design.

As such, the proposed scenario examines five stages of separative depth-wise CNN for golf swing feature extractor which is shown in Figure 6. Specifically, there are four

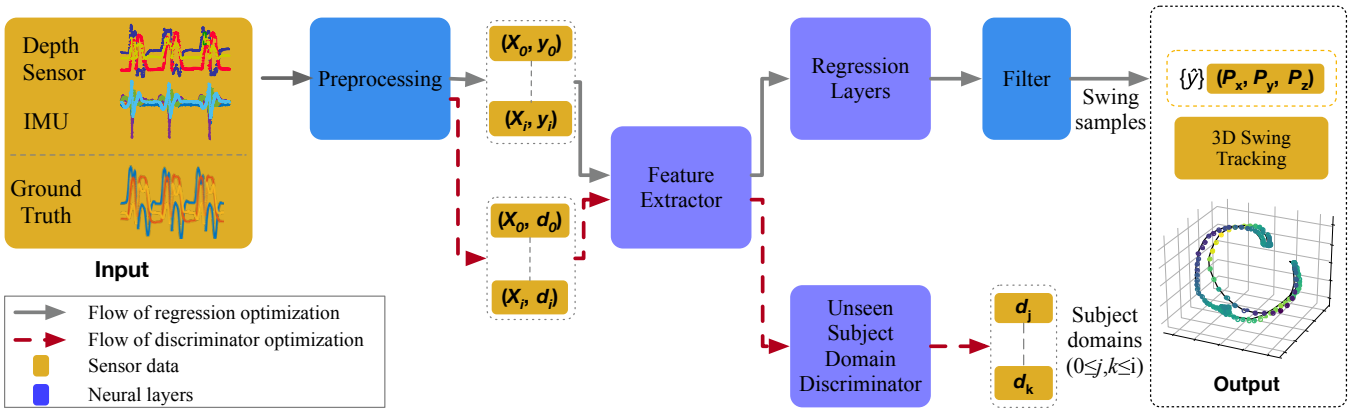

Figure 3. ***Condor*** **system Overview. Input: concatenation of the IMU and Depth sensor, denoted as X_i. Ground truth: denoted as y_i. Subject index: labeled as the domains as d_i. The framework trains two turns in each epoch. 1. The regression path learns a relationship from the fusion of the depth and IMU sensor to the ground truth (see Section 3.2 for details). 2. The discriminator path learns the ability that the model cannot discriminate a swing belongs to which subject (see Section 3.3 for details). The two networks (i.e., regression and subject classification) gradually reach a balance during training, which improves the feature extractor. The prediction only use the regression path to generate a batch collection of 3D swings, denoted as $\hat{\mathbf{y}}$.**

components in the feature extractor. Firstly, we use a convolutional network, i.e., Conv 3×3 in the figure, to encode the sensor measurements from different modalities to the input format of the next component, i.e, depth-wise Separable Convolution (DSC). Secondly, we make use of the DSC to fast filter the sensor signals and group or domain features of a swing. As a result, the learned grouped convolutions can extract the interrelationships between the IMU and depth sensor measurements during the swing. Thirdly, the learned features are shuffled to make the model robust. Finally, a Point-wise Convolution (PC) layer, i.e., Conv 1×1 just before the decoder, is inserted to convert the shuffled features into the input format of the decoder.

Since the dimension of depth sensor measurement is 3 (Kw_x, Kw_y, Kw_z), and that of IMU is 6 $(Acc_x, Acc_y, Acc_z, Gyr_x, Gyr_y, Gyr_z)$, the dimensions of the input to the feature extractor of *Condor* in a swing after segmentation in Section 2.3 are $\mathbf{K} \in \mathbb{R}^{3\times T}$ (depth sensor) and $\mathbf{I} \in \mathbb{R}^{6\times T}$ (IMU) respectively. In *Condor*, we use concatenation to combine different input sensor modalities (i.e., IMU and depth sensors), and denote it with square brackets. As such, the input can be written as:

$$\mathbf{X} = [\mathbf{K}, \mathbf{I}] \tag{1}$$

In *Condor*, each filter of CNN performs dot production in a small portion of input data for each layer to compose features. This process is followed by an activation function *ReLu*, which produces a summation of the dot products when they are positive, or 0 otherwise. Based on the convolution of these portions of data, each filter is replicated across the entire input data, and produces a rectified feature map. For convolutional layer l, filter i, this process can be viewed as:

$$P_l^i = Relu(BN(\sum \omega_l^i P_{l-1}^i + b_l^i)), \tag{2}$$

where *BN* is the Batch Normalization [17], and ω_l^i and b_l^i represents the weight and bias parameters respectively, which will be learned during the training process. The input is X for the first convolutional layer (i.e., $P_1 = X$) and the output (feature map) of previous layers for the other layers.

The CNN kernel sizes are shown in the feature extractor part of Figure 6. A pooling layer is applied to the output P_l^i of the first and last CNN layers to reduce data dimensions. We used max polling and global average polling in the first and last layers respectively (the black rectangles in the feature extractor of Figure 6).

Condor has three domains (Spatial, Modality and Temporal, see Figure 6) in part of the extractor. Here, we denote j as the index of domain, and Eq.(2) becomes:

$$P_l^{ij} = ReLu(BN(\sum \omega_l^{ij} P_{l-1}^{ij} + b_l^{ij})). \tag{3}$$

3.2 Regression layers

For swing tracking regression in a 3D space, the decoder of *Condor* makes use of a LSTM neural network. When dealing with time-series data involving 3D-based point tracking, LSTM provides a powerful type of recurrent neural network (RNN) for sequential data regression, especially for IMU measurements with errors accumulated over time [25]. Essentially, this is because RNN can model both short and long interrelationships within fixed measurements, which is challenging for other methods. Besides, LSTM has also addressed the long-term dependency problems in traditional RNNs because of gradient descent of an error criterion by introducing a sophisticated repeating structure.

As such, a common LSTM fully connected convolutional (FCN) neutral network was selected as the decoder in *Condor* because it is designed for time series data regression [20] and produces better performance in most related work than its variations [15, 20]. Meanwhile, one recent study with a time series datasets consisting of 3,627 experiments [21] saw that an LSTM-FCN neutral network outperforms other models such as FCN or attention-based LSTM-FCN. Our ablation study in Table 5, Section 4.5 demonstrates its effective-

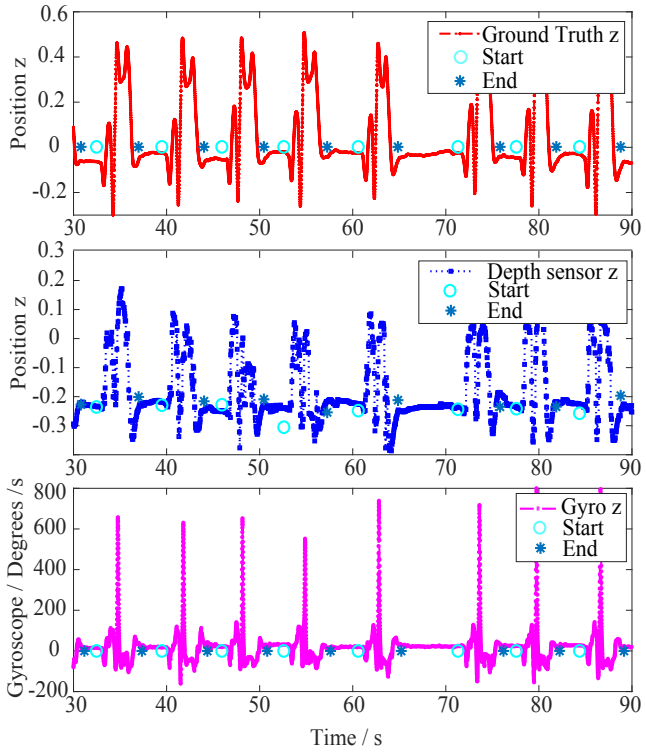

Figure 4. Signal synchronization and segmentation. O and $*$ denote the start and endpoint of a swing, respectively. Top: the z-axis of Ground Truth; Middle: the z-axis of depth sensor measurements; Bottom: the z-axis of gyroscope measurements. Note that the signal of one axis must be segmented as the measurements of different axes (as well as those of the accelerometer) are synchronized. The Ground Truth and depth sensor are aligned with the marker on depth sensor.

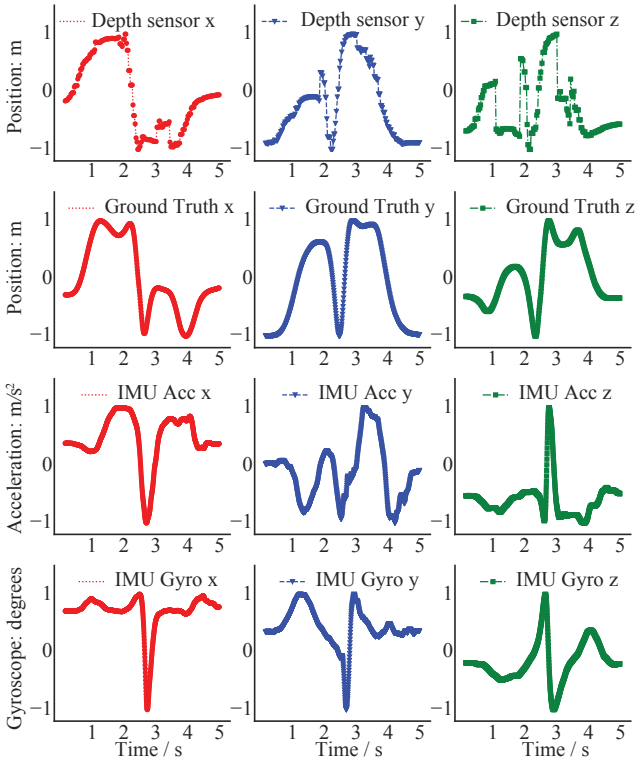

Figure 5. The measurements of different sensor modalities in one swing motion after pre-processing.

ness in regression against its variations such as Bidirectional LSTM (Bi-LSTM).

3.2.1 *LSTM as regression layers*

Regression layers is shown in Figure 7. LSTM has four gates and one cell state. For the purpose of clarity, we denote the output of the (last CNN layer of) generator of regression P_{last} as P, which is also the input of the decoder LSTM. Let w_i denote the weight and β_i denote the bias for an input gate respectively, which will be learned in the training. Then, the input gate (i_t) can be calculated as:

$$i_t = \sigma\left(w_i\left[h_{t-1}, P_t\right] + \beta_i\right), \tag{4}$$

where t is a time stamp, h_{t-1} is the hidden state in previous time stamp, and σ is the sigmoid function. Similarly for forget gate, w_f and β_f are the weight and the bias of a forget gate to be learned respectively. The forget gate (f_t) is:

$$f_t = \sigma\left(w_f\left[h_{t-1}, P_t\right] + \beta_f\right). \tag{5}$$

Let w_o and β_o denote the weight and bias of an output gate to be learned respectively. Then, the output gate (o_t) is:

$$o_t = \sigma\left(w_o\left[h_{t-1}, P_t\right] + \beta_o\right). \tag{6}$$

Let $\tilde{c}_t$ denote update gate as:

$$\tilde{c}_t = \tanh\left(w_c\left[h_{t-1}, P_t\right] + \beta_c\right), \tag{7}$$

where w_c is the weight and β_c is the bias of update gate to be learned respectively. Cell state c_t can then be calculated as:

$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t. \tag{8}$$

Finally, the output of the LSTM h_t can be calculated from:

$$h_t = o_t * \tanh\left(c_t\right). \tag{9}$$

Here, we can see that LSTM can take both current input and previous context/states into account for regression.

3.2.2 *Filtering*

In the prediction, we found that the trajectory of our network changes abruptly around the corresponding points in the ground truth. To this end, we use a Savitzky−Golay (S−G) filter to smooth the time series predictions, i.e., the Filt(er) Block in Figure 7. This procedure can seem as post-processing as its function is to smooth the regression prediction and we have added the S−G filter to all methods in our evaluation in Section 4. The S−G filter can be formally defined as:

$$Y_j = \sum_{i=\frac{1-m}{2}}^{\frac{m-1}{2}} C_i y_{j+i}, \quad \frac{m-1}{2} \le j \le n - \frac{m-1}{2}, \tag{10}$$

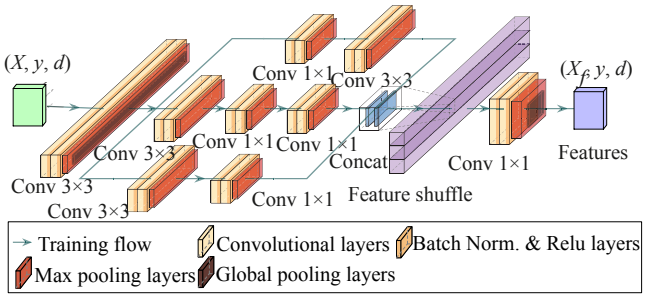

Figure 6. The structure of *Condor* deep learning framework. The input of the model is the concatenation of the IMU and depth sensor (Kw_x, Kw_y, Kw_z, Acc_y, Acc_z, Gyr_x, Gyr_y and Gyr_z) in T time stamps. The feature extractor uses depth-wise separable convolutions to capture information and can also make it lightweight. Time stamps T could be any. In our experiment, $T \geq 150$ is good enough for a smooth representation.

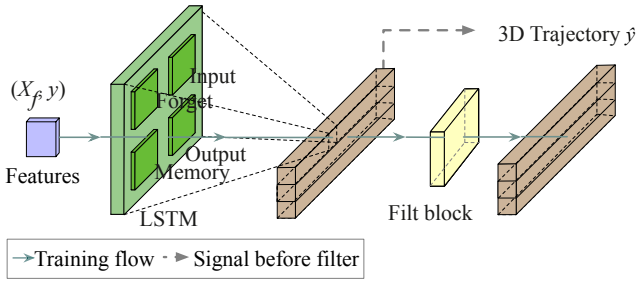

Figure 7. Generator as regression layers. The input is the encoded features from the feature extractor. The output is the trajectory of the point being tracked (Pos_x, Pos_y, Pos_z) in T time stamps.

where m is the convolution coefficients, $j \in n$ represents the j^{th} smoothed data points, and C is a constant parameter based on m and j. In *Condor*, the m and j are chosen as 13 and 3, respectively.

3.3 Subject-independent discriminator

We designed *Condor* for two scenarios: subject dependent and independent. Here, the 'domains' are referred to as the subjects. The 'domain adaptation' refers to remove the subject dependence. The subject-independent model is designed for unseen-subject swing tracking to alleviate the burden of data collection, labeling and supervised training. In this case, the player can directly use *Condor* system to perform swing tracking without training. Inspired by [19], we design a discriminator to make the CNN-LSTM model to perform swing tracking for unseen subjects.

The discriminator is shown in Figure 8. It takes the encoded features and labeled tags as input. The learned tags are the output. The discriminator is optimized with Stochastic Gradient Descent (SGD) optimizer, with 0.9 momentum and 5e-5 weight decay. The discriminator learns the real tags of the labeled domains (i.e., subjects).

In out experiment setting, the features are encoded as a format of batch multiple 456, followed by three linear layers with input features of 456, 128, and 32, respectively. The

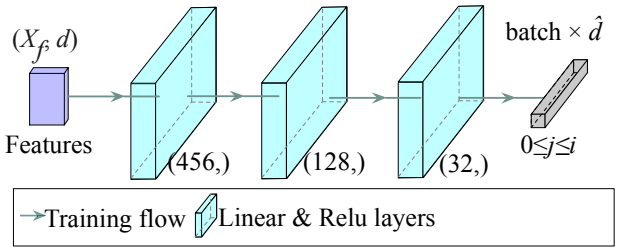

Figure 8. Subject independent discriminator. After train the extractor and regression path, the encoded features shares the regression ability but still bears the subject-dependent prior. We take the subject domains which are labelled with sensor data as the discriminator label to train the discriminator. The two-game training reaches a balance that the model is both able to perform regression and cannot discriminate which subject the data are labelled. By designing so we can achieve a subject-independent model.

output features of linear layers are 128, 32, and 12, respectively. The weights of all layers in discriminator are initiated with normal initialization with 0 mean and 0.1 standard deviations. All biases are initiated with 0.

3.4 Optimization

The changes of the depth sensor measurements in *Condor* are often abrupt because of occlusions, especially in stages such as top of swing and impact (see Figure 2). Furthermore, the IMU sensors in *Condor* also have acceleration effects (i.e., the changes in the sensor bias depending on how the sensor experiences acceleration) during different stages of the swing. Therefore, we choose Huber loss function as part of the loss function of *Condor* because it is less sensitive to outliers, compared to the other loss functions such as Mean Square Error (MSE) [12]. The Huber loss of regression prediction vector ($\hat{y}$) and ground truth (y) with dimension of T time stamps (note that y_i and $\hat{y}_i$ are 3D points) is formally defined as:

$$\mathcal{L}_h(y,\hat{y}) = \frac{1}{T}\sum_{t=1}^{T} z_t \tag{11}$$

wherez_t is given by:

$$z_t = \begin{cases} 0.5\,(y_t - \hat{y}_t)^2, & \text{if } |y_t - \hat{y}_t| < 1 \\ |y_t - \hat{y}_t| - 0.5, & \text{otherwise.} \end{cases} \tag{12}$$

For swing tracking, the trajectory defines a plane in 3D space for the swing, which is the key information that defines which direction and speed the ball goes after impact. Therefore, we add this information into the loss function of *Condor* by calculating the angle between the planes defined by y and $\hat{y}$ respectively. Specifically, we use Cosine similarity to calculate the difference between two planes, which is defined as:

$$\mathcal{L}_p = 1 - \cos(\theta) = 1 - \frac{\sum_{i=1}^{T} y_i \hat{y}_t}{\sqrt{\sum_{t=1}^{T} y_t^2}\sqrt{\sum_{t=1}^{T} \hat{y}_t^2}} \tag{13}$$

In domain discriminator, the true domains are labelled as

Table 1. Prototype configuration

Items	Description
IMU	Samsung Gear S
IMU Sampling Rate	100 Hz
IMU Signals	3-axis acceleration and 3-axis gyroscope
Depth sensor	Realsense D435
Depth Sensor Sampling Rate	30 fps
Depth Sensor Signals	3-axis locations of a wrist
Ground Truth	OptiTrack
OptiTrack Sampling Rate	120 fps
Subject distance from depth sensor	1 - 3 m

d, the learned domains are labelled as $\hat{d}$. We use cross entropy loss to learn the discrepancy between the labelled domains and true domains. Thus the domain discrimination loss can be denoted as:

$$\mathcal{L}_d = -d \cdot \log(\hat{d}). \tag{14}$$

In consideration of the loss in GAN setting, where we wish discriminator cannot tell which domain comes from, we multiply a negative sign in domain loss and also applying a regulation parameter α ($0 \leq \alpha \leq 1$, in our experiments we select $\alpha = 0.6$) to balance the first two losses. Combining Eq. (11), (13), (14), the total loss function thus becomes:

$$\begin{aligned}\mathcal{L}(y,\hat{y}) = \alpha \times \mathcal{L}_h(y,\hat{y}) + (1-\alpha) \times \mathcal{L}_p(y,\hat{y}) \\ -1 \times \mathcal{L}_d(d,\hat{d}).\end{aligned} \tag{15}$$

4 Experiments

4.1 Implementation

We design and implement a *Condor* prototype based on a Realsense D435 depth sensor and a Samsung Gear S smart wristband (IMU). The ground truth in this paper was collected using a motion-capture system comprising 25 high-fidelity cameras called OptiTrack [32] on the ceiling (see Figure 9). These cameras operate at a 120 fps sampling rate for marker-based tracking and can achieve millimeter-level tracking accuracy in 3D. The detailed prototype configuration is shown in Table 1.

4.1.1 Setup

The IMU sensor and a marker for OptiTrack are attached to one of the wrists of a user (depending on the user's preference) as the point of interest being tracked (see Figure 9). The raw depth signal from the depth camera produced 25 of the most important skeleton joints through feature extraction, with model fitting by Nuitrack API. Meanwhile, the 3D position of the skeleton joint corresponding to the IMU sensor's location on the wrist was selected as one of the input variables in the deep neural network of *Condor*. One alternative is to choose the end of the club as the swing-tracking point; however, this means the IMU becomes a target that inevitably collides with the ball during impact. Evidently, this significantly reduces the device's life span over time.

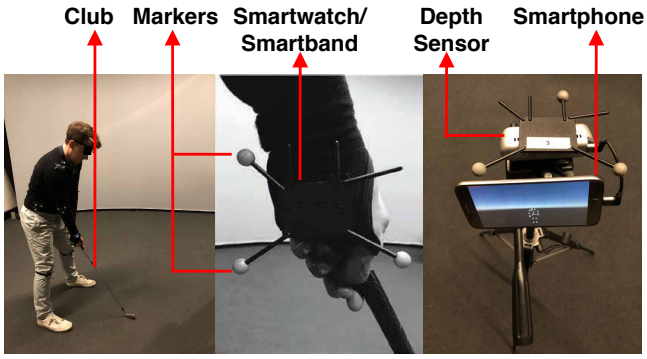

Figure 9. Setup of the proposed *Condor* framework. Analysis was completed by placing an IMU sensor on a subject's wrist and a depth sensor face-on. Sensors in proposed framework are all ubiquitous and mobile.

4.1.2 Data collection

We collected our datasets based on 12 subjects (10 males and 2 females)[2], including 3 professional golf players to collect the data. Their age is ranging from 26 to 35, height from 160 to 178 cm, and weight from 52 to 74 kg. During the data collection, in each swing, we will shut down all IMU, depth sensor, and Ground Truth when the subject feels tired or want to stop.

We have 12 sensor modalities, i.e., a three-axis of depth sensor location (i.e., Kw_x, Kw_y and Kw_z defined in Section 2), accelerometer (Acc_x, Acc_y, and Acc_z), gyroscope (Gyr_x, Gyr_y and Gyr_z) and ground truth location (Pos_x, Pos_y and Pos_z) produced by OptiTrack, respectively, and chose the fixed number of time stamps defined in Section 2 as $T = 152$ in one swing-motion segment according to the sampling rates of the sensors specified in Table 1. As such, the total size of the dataset contained more than 2 million sensor measurements. Furthermore, the data were collected over a period of 4 weeks and with different D435 sensor, Samsung Gear S, and golf club hardware because the golf swing is an energy-intensive exercise and long-term continuous swings could cause muscle fatigue.

4.1.3 Evaluation Metrics

For subject-dependent model, we take 80% of all collected data as training set and 20% as evaluation. For subject-independent model, we take data collected from 11 subjects as training data and the remaining unseen subject as test data. A more detailed discussion about the dataset for subject-independent model is discussed in Section 4.6. We use an average of a 10-fold cross-validation for these evaluation. The principal goal was to evaluate the accuracy of 3D position tracking, which has T time stamps. Here, T can be any value while our test find out that $T \geq 150$ is good enough for a smooth swing (we choose 152 for convenience of model parameters selection). Thus, our main evaluation metrics for analysis include both the **Root Mean Squared**

[2]Data collection involving human subjects has been approved by the ethics committee of the corresponding organization (HC17818).

Error (RMSE) as

$$\frac{1}{T}\sum_{i=1}^{T}\sqrt{(Pos_x^i-\hat{Pos}_x^i)^2+(Pos_y^i-\hat{Pos}_y^i)^2+(Pos_z^i-\hat{Pos}_z^i)^2}, \quad (16)$$

and **Mean Absolute Error (MAE)** as

$$\frac{1}{T}\sum_{i=1}^{T}(|Pos_x^i-\hat{Pos}_x^i|+|Pos_y^i-\hat{Pos}_y^i|+|Pos_z^i-\hat{Pos}_z^i|). \quad (17)$$

Table 2. Subject dependent 3D position tracking performance. ** denotes the state-of-art BiLSTM model using IMU only called IONet, while * denotes using sensor fusion. For fair comparison, we compared BiLSTM for both methods.

Model	MAE (cm)	RMSE (cm)
IONet (BiLSTM)**	11.13 ± 0.83	16.98 ± 0.93
SVR*	29.71 ± 2.36	35.58 ± 4.12
LSTM*	10.17 ± 0.55	14.54 ± 0.81
BiLSTM*	8.95 ± 0.53	12.97 ± 0.80
Condor*	$\mathbf{4.71 \pm 0.30}$	$\mathbf{6.57 \pm 0.60}$

Table 3. Subject independent 3D position tracking performance

Model	MAE (cm)	RMSE (cm)
Without domain adaptation	9.43 ± 0.68	$\mathbf{11.50 \pm 0.91}$
With domain adaptation	7.37 ± 0.54	$\mathbf{9.38 \pm 0.77}$

4.2 Overall performance

Baselines. IONet [7] is the state-of-the-art model for IMU-based tracking. The IONet prediction has three different neutral network models, i.e., CNN, LSTM and BiLSTM, and the BiLSTM produces the best performance in our experiment. Furthermore, we have produced the model of BiLSTM for both IMU-based prediction and sensor fusion (i.e., IMU plus depth sensor) prediction in Table 2. Firstly, it shows that using a BiLSTM model, adding depth sensor modality (termed BiLSTM in Table 2) improves tracking accuracy, unsurprisingly (i.e., RMSE reduced from 16.98 cm to 12.97 cm). Secondly, *Condor* has re-modelled the neural network architecture, which further improves tracking performance (RMSE: 6.57 cm). Finally, *Condor* features a cGAN architecture that can work with unseen subjects (see Table 3).

We also compares other baselines include Support Vector Regression (SVR) and LSTM. Table 2 shows how *Condor* performs best by reducing tracking errors by approximately 50% compared to the second-best (i.e., BiLSTM). For example, the RMSE for BiLSTM is 12.97 cm while that of *Condor* is only 6.57 cm. It reduces tracking errors by approximately 82% and 61%, respectively, compared to SVR and IONet. We observe the better performance produced by LSTM and BiLSTM than that of IONet because they make use of both depth sensor and IMU information, while IONet relies on IMU only. We observe similar results for MAE likewise.

Figure 10 shows some randomly chosen swings along the z-axis. Evidently, occlusions occur regularly during swing track, and these have a great effect on the performance of the depth camera. This certainly justifies our decision to pair sensor fusion with an IMU and a depth camera. The figure also shows that the *Condor* produces very good quality tracking results which are very close to ground truth, and significantly outperforms the state-of-the-art deep learning-based tracking system IONet.

Figure 10 also shows that *Condor* has calibrated different types of errors in swing tracking. Specifically, state-of-the-art methods sometimes predict wrong directions due to occlusions or accumulated errors at various stage changes. For example, as shown in Figure 10, IONet offen made error at the Stage Top of Swing, but *Condor* corrected these anomalies successfully.

The subject-independent performance in Table 3 shows that with and without discriminator as domain adaptation path has an approximately 18.4% improvement. A further detailed discussion towards domain adaptation is discussed in Section 4.6.

4.3 Impact of Different Sensor Modalities

We study how *Condor* learns multi-modality representations during one swing among depth sensor position, acceleration and gyroscope, respectively. To this end, we activate one sensor modality only by setting the measurements of the rest sensor modalities to zero for each swing. Then we input such augmented measurements to the *Condor* model to calculate a position prediction based on each sensor modality only in turn in each timestamp. Finally, we compare the predictions and ground truth to find the sensor modality that produces the minimum error in each timestamp. We group the sensor measurements into five stages as shown in Figure 2, and investigate the ratio of different sensor modalities that produce a minimum error in each stage. Figure 11 shows the results of the analysis. Unsurprisingly, in Stages 1 (Setup) and 5 (Finish), the depth camera plays a key role when the speed of the tracking point is relatively slow and occlusion does not occur. The contribution (the ratio of minimum error) of the depth sensor gradually decreases in Stage 2 (Top of Swing). When the speed intensifies (up to 44.7 m/s) and occlusion is likely, e.g., in Stages 2, 3 (Impact) and 4 (Follow Through), the IMU plays an increasingly important role. Note that the depth sensor produces less importance in Stages 3 and 4, and take near all dominance in Stages 1 and 5. Figure 12 shows the sensor modality that produces a minimum error for a sample swing (x axis) and further illustrates their contributions over time. The figure shows that *Condor* is further able to learn the temporal dynamic characteristics of different sensor modalities to produce highly accurate tracking results.

4.4 Impact of Loss Function

We investigated the effects of the different loss function in *Condor*'s deep learning model to measure 3D tracking accuracy in the golf swing. Table 4 shows that, for single loss functions, the performance of Huber is the best by producing RMSE and MAE of 6.82 cm and 5.05 cm, respectively. However, the performance can be further improved by the

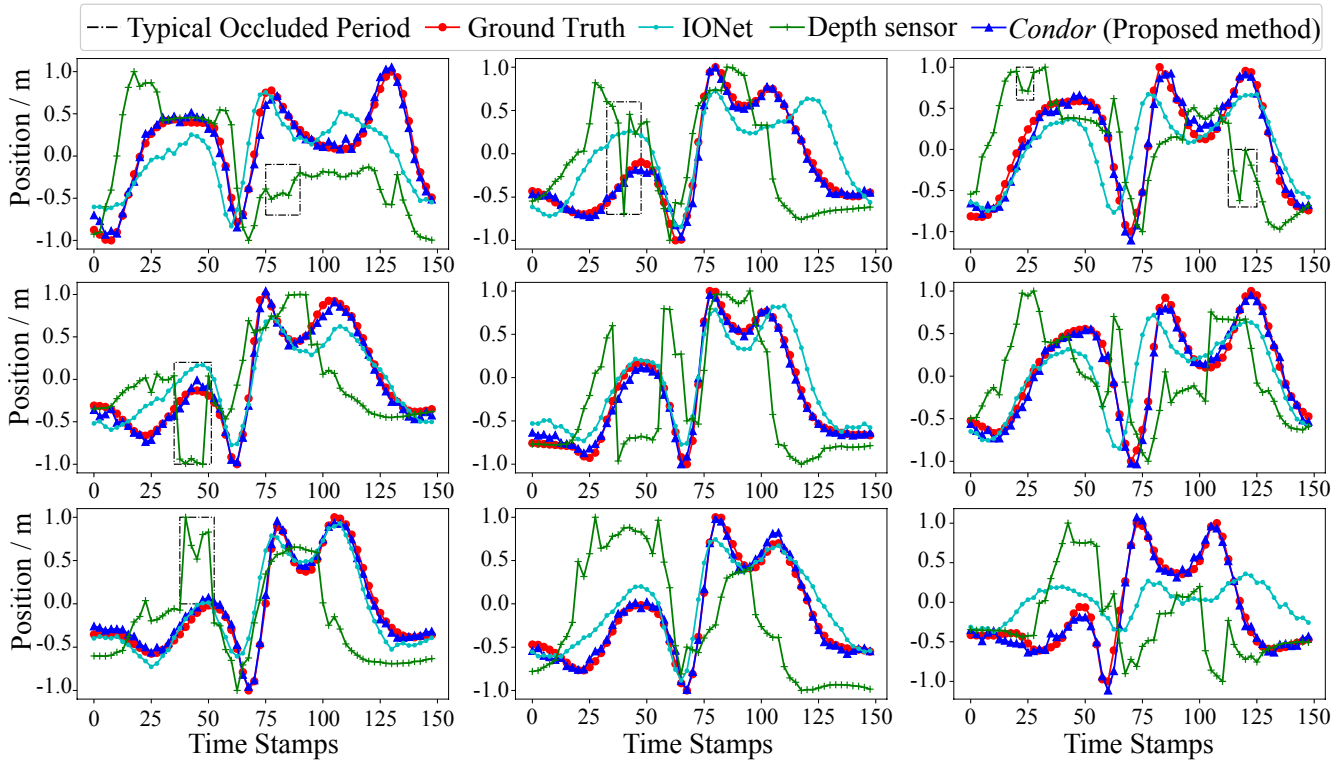

Figure 10. Different patterns of swings calibrated by *Condor*. It is clear that during the swing, depth sensor bears many occluded errors. By fusing these two sensors, proposed method can calibrate the occlusion and close the swing prediction to the ground truth. IMU-based solution, although may catch an trend of swing based on deep learning, it still cannot predict precisely due to its mechanism where IMU are easily cause cumulative error.

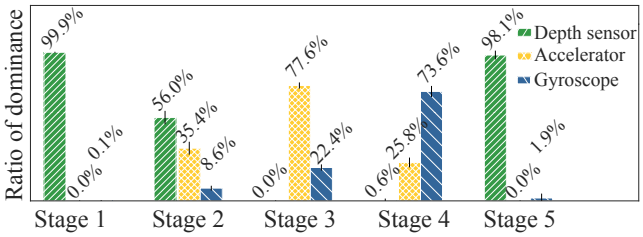

Figure 11. Ratio of contributions of different sensor modalities over five stages of a swing. Not surprisingly, depth sensor offers more accurate tracking in Stage 1 and 5. Gradually, however, IMU plays an more important role to calibrate depth camera when it is occluded. An interest trend is that Gyroscope gradually dominate the tracking task during vision-occluded stages.

combination of Huber and Cosine distance (Eq. (15) without domain discriminator in Section 3.4) to 6.57 cm and 4.71 cm, respectively. Also, compared to common loss functions MSE and MAE, the proposed loss function reduces MAE by approximately 32% and 10% respectively. The other metric RMSE also shows similar patterns. We note that the value of regulation parameter α in Eq. (15) was learned to 0.6 automatically by our model during training.

Table 4. Comparison of different deep learning loss functions

Model	MAE (cm)	RMSE (cm)
MSE	6.17 ± 0.30	8.09 ± 0.28
MAE	5.16 ± 0.40	7.05 ± 0.34
Cosine Distance	16.40 ± 1.90	35.60 ± 2.40
Huber	5.05 ± 0.20	6.82 ± 0.30
Condor	$\mathbf{4.71 \pm 0.30}$	$\mathbf{6.57 \pm 0.60}$

4.5 Impact of Model Parameters

We investigated the effects of the numbers of LSTM or BiLSTM layers in the proposed *Condor* LSTM architecture to its performance. Table 5 shows that 128 layer LSTMs produce the best result. Although the BiLSTM has double parameters as those of the single-layer LSTM, it does not produce significantly better tracking accuracy. Therefore, *Condor* chooses 128 single-layer LSTM as the decoder to reduce resource consumption in mobile devices.

4.6 Impact of Domain Adaptation

4.6.1 Domain adaptive path

The domain-adaptive model is designed specifically for unseen-subject scenarios. The intuition for the domain-adaptive path is to improve the original CNN-LTSM model to generate swing tracking for unseen subjects. To evaluate

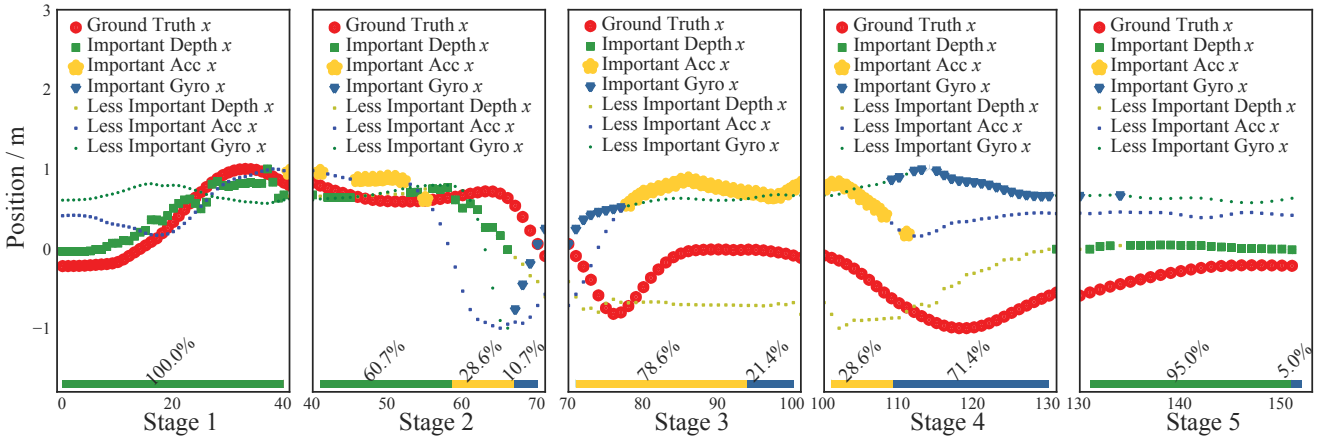

Figure 12. Qualitative analysis of the dominant measurements (x axis) over time for one example swing for *Condor* visualization. In the first stage (i.e. Set-up), depth sensor ■ (green) is more close to the ground truth (GT) ● (red). In stage two, when depth camera is occluded, the Gyroscope ▼ (blue) tries to pull up the tracking to the GT. In stage tree and four, Gyroscope and Acceleration ⬟ (yellow) from IMU shares more common trend than other depth sensor. In stage five, depth sensor again close to the GT more than IMU.

Table 5. LSTM parameter selection. lr: layers. Bi: BiLSTM

Metrics	64lr	128lr	256lr
MAE (cm)	5.04 ± 0.19	$\mathbf{4.71 \pm 0.30}$	5.32 ± 0.28
RMSE (cm)	6.88 ± 0.30	$\mathbf{6.57 \pm 0.60}$	7.26 ± 0.29
Metrics	64lr Bi	128lr Bi	256lr Bi
MAE (cm)	5.10 ± 0.22	4.73 ± 0.35	4.96 ± 0.15
RMSE (cm)	6.90 ± 0.32	6.62 ± 0.28	6.58 ± 0.23

the domain adaptation, we use the original CNN-LSTM as baseline. Table 3 shows that the original regression path can be improved via domain adaptation by 18% in RMSE.

4.6.2 *Training size of domains*

In a domain-adaptive scenario, the *Condor* may be trained with different numbers of domains (i.e., subjects). We evaluate the impact of domain size by changing different numbers of domains in the training set. Figure 13 shows that the increase of domains can improve swing tracking accuracy. With 11 domains being trained, the model reaches the lowest RMSE with 9.38 cm.

4.7 Micro Edge Benchmark

We measure the resource and computation consumption of *Condor* swing tracking prediction on a popular smartphone platform iPhone X [3] with Core ML using iOS Energy Diagnostics profiling template. We repeat the prediction for 1,000 times and report the average values. Table 6 shows that it takes 4.56 ms only for the ML Core to make a prediction, which enables realtime 3D golf swing tracking feedback. The energy consumption of *Condor* is only 152 mJ for one swing prediction. Thus, 1% of iPhone X's battery (i.e., 2,716 mAh) can support more than 590 *Condor* swing prediction. The deep learning model of *Condor* has 90,300 parameters and consumes 0.6 Mega floating-point operations per second (MFLOP), which only further demonstrates *Condor*'s resource efficiency.

[3] Model: NQA62ZP, Core ML 3, ios 13.0

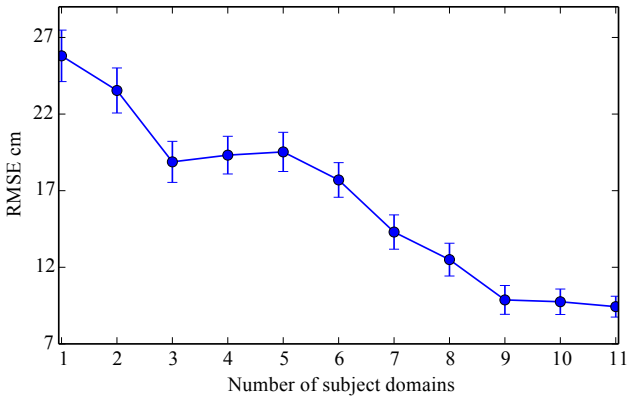

Figure 13. RMSE under different number of unseen subject domains. Clearly, with more domains been collected, the tracking accuracy will be further improved.

Table 6. Resource consumption (prediction)

Model	Times (ms)	Energy (mJ)	Number of Parameters (K)	MFLOP (M)
Condor	4.56	152	90.3	0.6

5 Discussion and Future Work

Indoor experiments. The aim of the *Condor* is to provide golf swing tracking and analytics in real-world settings, such as an outdoor golf course. There, dynamic outdoor lighting conditions may negatively affect the performance of depth sensors and, in turn, tracking reports. This means the results reported for *Condor* may have been favorable to the otherwise stable lighting conditions found indoors. Lack of an

outdoor ground truth system remains a limitation to which we are exploring alternatives.

Stereo cameras. Multiple (dual) cameras and LiDAR are available in many latest smartphone models such as iPhone, Galaxy S9 and P30 Pro, which could be used to render objects in 3D depth maps similar to depth cameras. We believe that the principles of *Condor* can be applied to these smartphones directly. The challenges here are how to produce important skeleton joints efficiently as we did for depth cameras in Section 4.1.

Generalization. We believe that the principle of *Condor* can be applied to the swing tracking in other sports, such as baseball and tennis. However, to evaluate the performance of *Condor* in these sports will involve non-trivial extra data collection, which we will leave as future work.

6 Related Work

Swing tracking research can be broadly divided into two categories: vision-based and IMU-based. Vision-based swing tracking mainly focused on pose estimation [1, 2]. One example concerns benchmark OpenPose, which is a real-time multi-person key-point library that can convert the subjects in 2D images into 3D skeleton joints. However, 2D to 3D estimation is still ambiguous [30]. Also, to reach an occlusion-free tracking in mobile, multiple video camera is extremly inconvenient for end users [27].

Similarly, stacked Hourglass Networks [29] produce a CNN architecture for human pose estimation. However, the model is computationally expensive and time-demanding. Another problem for stacked Hourglass and all other vision-based pose estimation solutions is occlusion. During a swing, a player's wrist may be hidden behind their body, especially in the Stages Top-of-Swing and Finish (see Figure 2), because the field of the view of the camera will be blocked if the camera is placed in front of the player. Similarly, occlusion will happen in other stages if the camera is placed in other locations instead of in the front.

By subsequently leveraging such depth cameras as Kinect, Yu et. al. proposed to produce a 3D spatial trajectory via a golf club [35]. This system uses depth images and the depth-related shadow to calculate the 3D trajectory of the club. By establishing the spatial trajectory, this model can eliminate the need of the IMUs, which must otherwise be attached to the subject's body (i.e., their wrist) or their club. Nonetheless, occlusion remains a barrier and successful operation only extends to indoor environments due to the comparatively complex lighting conditions found outdoors. Kumada et. al. perform a pixel level analysis on swings [23], but is insufficient for informative swing analysis as a pixel to 3D is ambiguous. Similarly, Soonchan et. al. proposed a multiple random forest tree method to vote and cast the occluded wrist joints called SWAN [30]. However, as the authors concluded, SWAN often misses some key phases due to fast swings that cannot be captured with Kinect.

The other category of swing tracking is based on IMUs or other multiple sensor modalities [4, 22, 37]. Shen et. al. explores the IMU tracking to eliminate the accumulated error for motion tracking [34]. However, this work reaches the accuracy of IONet and still suffers from the noise model of IMU. Chesser et.al using a mobile BLE device with a beacon device to do indoor tracking [8]. However, the 23.5 cm accuracy makes it very hard to deploy for fast swing tracking. Nam et. al. proposed a system that utilizes an IMU and stereo cameras to track a golf club [28]. For this, a swing's motion-tracking algorithm was proposed to obtain the club's orientation through a T-shaped LED. However, using one-third length of the club's LED as the marker cannot offer a practical prediction. The system also needs to be practiced indoor to fix the camera and suffers a 13.2 cm error. Other works also make use of multiple IMU sensors combined with sophisticated sensor fusion methods [3, 5]. S. Chun et. al. attached two IMUs on a player's wrist and their club, respectively, to calculate the trajectory of the motion [10]. However, these systems proved both immobile (e.g., the need for a bino-camera [28]) and low accuracy. Recently, Chen et. al. proposed a deep learning-based method (IONet) to calibrate the IMU sensors [7], which inspired our work in *Condor*. Importantly, this paper contends to significantly expand the deep learning model using a novel sensor-fusion architecture that produces significantly better swing-tracking performance.

In addition to the academic research works, there are some commercial mobile (golf) swing tracking and analysis products on the market such as Zepp[4], Rapsodo[5], PIQ Golf Sensor[6], Blast Motion[7] and Garmin TruSwing[8]. However, IMU-based products such as Zepp cannot not show 3D swing tracking trajectory, which lost one of the main information for sports analytics [14]. As discussed, depth sensor-based products cannot accurate render the player's swing motions in 3D space.

In contrast, *Condor* attempts to explore the 3D depth-sensing capabilities of the latest mobile technologies such as Tango-enabled smartphones and Intel RealSense depth cameras to improve the 3D tracking accuracy of swings.

7 Conclusions

This paper proposed a lightweight golf swing-tracking CNN-LSTM model that fuses measurements from different sensor modalities and further addresses the challenges in single-sensor modalities, including occlusion, low sampling rates, and complex sensor noise models. As shown, the model, known as *Condor*, can be deployed in ubiquitously available smart wristbands and smartphones to provide IMU and depth sensor measurements, respectively. Our extensive evaluation shows that *Condor* produces highly accurate tracking performance and can reduce tracking errors by approximately 62% compared to state-of-the-art approach. For unseen subject scenarios, our proposed conditional GAN in *Condor* can achieve approximately less than 10 cm tracking accuracy.

8 Acknowledgments

This research is supported by Australian Government Research Training Program (RTP) Scholarship.

[4]Zepp 2020. https://www.zepp.com/us/zepp-e.

[5]Rapsodo 2020. https://rapsodo.com/golf/rmotion.

[6]PIQ Golf Sensor 2020. https://piq.com/golf.

[7]Blastmotion 2020. BlastMotionGolfReplayAnalyzer.

[8]TruSwing 2020. https://buy.garmin.com/en-US/US/p/505083.

9 References

[1] R. Alp Güler, N. Neverova, and I. Kokkinos. Densepose: Dense human pose estimation in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7297–7306, 2018.

[2] M. Andriluka, U. Iqbal, E. Insafutdinov, L. Pishchulin, A. Milan, J. Gall, and B. Schiele. Posetrack: A benchmark for human pose estimation and tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5167–5176, 2018.

[3] D. Arsenault. *A quaternion-based motion tracking and gesture recognition system using wireless inertial sensors*. PhD thesis, Carleton University, 2014.

[4] P. Blank, B. H. Groh, and B. M. Eskofier. Ball speed and spin estimation in table tennis using a racket-mounted inertial sensor. In *Proceedings of the 2017 ACM International Symposium on Wearable Computers*, ISWC '17, pages 2–9, New York, NY, USA, 2017. ACM.

[5] A. Brennan, J. Zhang, K. Deluzio, and Q. Li. Quantification of inertial sensor-based 3d joint angle measurement accuracy using an instrumented gimbal. *Gait & posture*, 34(3):320–323, 2011.

[6] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh. Openpose: realtime multi-person 2d pose estimation using part affinity fields. *arXiv preprint arXiv:1812.08008*, 2018.

[7] C. Chen, X. Lu, A. Markham, and N. Trigoni. Ionet: Learning to cure the curse of drift in inertial odometry. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[8] M. Chesser, L. Chea, and D. C. Ranasinghe. Field deployable real-time indoor spatial tracking system for human behavior observations. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, pages 369–370. ACM, 2018.

[9] S. Choppin, S. Albrecht, J. Spurr, and J. Capel-Davies. The effect of ball wear on ball aerodynamics: An investigation using hawk-eye data. In *Multidisciplinary Digital Publishing Institute Proceedings*, volume 2, page 265, 2018.

[10] S. Chun, D. Kang, H.-R. Choi, A. Park, K.-K. Lee, and J. Kim. A sensor-aided self coaching model for uncocking improvement in golf swing. *Multimedia Tools and Applications*, 72(1):253–279, 2014.

[11] D. Comotti, M. Ermidoro, M. Galizzi, and A. Vitali. Development of a wireless low-power multi-sensor network for motion tracking applications. In *2013 IEEE International Conference on Body Sensor Networks*, pages 1–6. IEEE, 2013.

[12] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[13] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 2672–2680, Cambridge, MA, USA, 2014. MIT Press.

[14] M. Gowda, A. Dhekne, S. Shen, R. R. Choudhury, L. Yang, S. Golwalkar, and A. Essanian. Bringing iot to sports analytics. In *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, pages 499–513, 2017.

[15] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232, 2016.

[16] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[17] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[18] L. Jiang, S. Xiao, and C. He. Kinect depth map inpainting using a multi-scale deep convolutional neural network. In *Proceedings of the 2018 International Conference on Image and Graphics Processing*, pages 91–95. ACM, 2018.

[19] W. Jiang, C. Miao, F. Ma, S. Yao, Y. Wang, Y. Yuan, H. Xue, C. Song, X. Ma, D. Koutsonikolas, W. Xu, and L. Su. Towards environment independent device free human activity recognition. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, MobiCom '18, page 289–304, New York, NY, USA, 2018. Association for Computing Machinery.

[20] R. Jozefowicz, W. Zaremba, and I. Sutskever. An empirical exploration of recurrent network architectures. In *International Conference on Machine Learning*, pages 2342–2350, 2015.

[21] F. Karim, S. Majumdar, and H. Darabi. Insights into lstm fully convolutional networks for time series classification. *arXiv preprint arXiv:1902.10756*, 2019.

[22] A. Khan, J. Nicholson, and T. Plötz. Activity recognition for quality assessment of batting shots in cricket using a hierarchical representation. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 1(3):62:1–62:31, Sept. 2017.

[23] K. Kumada, Y. Usui, and K. Kondo. Golf swing tracking and evaluation using kinect sensor and particle filter. In *2013 International Symposium on Intelligent Signal Processing and Communication Systems*, pages 698–703. IEEE, 2013.

[24] C. Li, Z. Zhao, and X. Guo. Articulatedfusion: Real-time reconstruction of motion, geometry and segmentation using a single depth camera. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 317–332, 2018.

[25] M. Li, H. Yu, X. Zheng, and A. I. Mourikis. High-fidelity sensor modeling and self-calibration in vision-aided inertial navigation. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 409–416. IEEE, 2014.

[26] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 116–131, 2018.

[27] N. Nakano, T. Sakura, K. Ueda, L. Omura, A. Kimura, Y. Iino, S. Fukashiro, and S. Yoshioka. Evaluation of 3d markerless motion capture accuracy using openpose with multiple video cameras. *bioRxiv*, page 842492, 2019.

[28] C. N. K. Nam, H. J. Kang, and Y. S. Suh. Golf swing motion tracking using inertial sensors and a stereo camera. *IEEE Transactions on Instrumentation and measurement*, 63(4):943–952, 2014.

[29] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision*, pages 483–499. Springer, 2016.

[30] S. Park, J. Yong Chang, H. Jeong, J.-H. Lee, and J.-Y. Park. Accurate and efficient 3d human pose estimation algorithm using single depth images for pose analysis in golf. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 49–57, 2017.

[31] A. Pfister, A. M. West, S. Bronner, and J. A. Noah. Comparative abilities of microsoft kinect and vicon 3d motion capture for gait analysis. *Journal of medical engineering & technology*, 38(5):274–280, 2014.

[32] N. Point. Inc.: Optitrack-optical motion tracking solutions, 2009.

[33] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[34] S. Shen, M. Gowda, and R. Roy Choudhury. Closing the gaps in inertial motion tracking. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, pages 429–444. ACM, 2018.

[35] T. Yu, Z. Zheng, K. Guo, J. Zhao, Q. Dai, H. Li, G. Pons-Moll, and Y. Liu. Doublefusion: Real-time capture of human performances with inner body shapes from a single depth sensor. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7287–7296, 2018.

[36] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6848–6856, 2018.

[37] H. Zhao, S. Wang, G. Zhou, and W. Jung. Tenniseye: Tennis ball speed estimation using a racket-mounted motion sensor. In *2019 18th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 241–252, April 2019.

[38] X. Zhou, M. Zhu, G. Pavlakos, S. Leonardos, K. G. Derpanis, and K. Daniilidis. Monocap: Monocular human motion capture using a cnn coupled with a geometric prior. *IEEE transactions on pattern analysis and machine intelligence*, 41(4):901–914, 2019.