透過關鍵點估計實現高精度羽球場地交點偵測之方法論框架

第一節：典範轉移：從邊界框到關鍵點的幾何特徵偵測

在電腦視覺領域，物件偵測的目標通常是定位並分類具有豐富視覺特徵的實體，如人、車輛或球拍。然而，當任務目標轉變為抽象的幾何圖形，例如羽球場地上的線條交點時，傳統的物件偵測方法便會遭遇其根本性的設計瓶頸。本報告旨在闡明，對於此類點狀幾何特徵的定位任務，關鍵點估計（keypoint estimation）不僅是一種可行的替代方案，更是一種在理論基礎和實踐效能上都更為優越的典範。

1.1 IoU 基礎偵測方法於點狀特徵的內在侷限性

標準的物件偵測框架，無論是YOLO、Faster R-CNN還是其他模型，其核心的定位評估與優化指標均為「交並比」（Intersection over Union, IoU）[1]。IoU透過計算預測邊界框（bounding box）與真實邊界框之間的面積重疊度來衡量定位的準確性。對於具有一定體積的常規物件，這是一個合理且有效的指標。然而，當應用於羽球場交點這類幾乎是點狀的特徵時，IoU的物理意義便開始瓦解。一份針對此問題的深入分析報告指出，一個理想的交點真實邊界框可能僅有幾個像素的大小 [1]。在這種極端情況下，預測框即便只有微不足道的幾個像素偏移，也可能導致IoU值從一個接近完美的值（例如0.9）災難性地驟降至0 [1]。這種現象揭示了IoU指標的根本性缺陷：它對點狀特徵的定位誤差極度敏感，卻無法提供有意義的梯度信號。

這個問題的根源在於，任務的真實目標（座標精確度）與模型的優化指標（面積重疊度）之間發生了根本性的錯位。模型的目標是找到交點的精確 (x,y) 座標，但基於IoU的損失函數（如CIoU、GIoU）卻在訓練模型解決一個不同的問題：「如何最大化兩個微小矩形的重疊面積？」這導致了幾個嚴重的後果：

1. 不穩定的梯度信號：當IoU從高值驟降至零時，損失函數的梯度會變得極其「陡峭」和嘈雜。模型在優化過程中，其預測可能在「幾乎完美」和「完全錯誤」兩個極端狀態之間劇烈震盪，難以平滑地收斂到亞像素級別的最佳定位點 [1]。
2. 缺乏方向性指引：當預測框與真實框完全不重疊時（IoU=0），儘管CIoU等改進版損失函數引入了中心點距離懲罰，但對於點狀特徵，這種信號依然很弱。損失函數無法有效地告訴模型應該「向哪個方向」移動幾個像素來修正錯誤。
3. 學習效率低下：模型被迫在一個極其困難的優化空間中進行探索。這個空間的損失地貌（loss landscape）可以被想像成一片廣闊的平原（代表IoU=0的區域），中間只有一個極其微小且陡峭的深坑（代表高IoU的區域）。優化器（如Adam或SGD）很難穩定地找到並停留在這個深坑的最低點。

因此，將標準物件偵測直接應用於幾何交點偵測，無異於用一把設計用來測量體積的工具去測量一個點的長度，其結果必然是粗糙且不穩定的。

1.2 關鍵點估計：一種幾何原生的解決方案

與之相對，關鍵點估計將問題的表述回歸其幾何本質。它不再試圖用一個「框」去包圍一個「點」，而是直接將任務建構為對關鍵點 (x,y) 座標的直接回歸 [1]。這種方法的優越性體現在其損失函數的設計上。

關鍵點估計任務的損失函數不再基於面積重疊，而是基於距離，例如簡單的L1/L2距離，或是更為先進的「物件關鍵點相似度」（Object Keypoint Similarity, OKS）。OKS是一種經過歸一化和尺度不變性處理的距離度量，其設計思想使其成為評估關鍵點定位精度的黃金標準 [4]。

採用這種基於距離的損失函數，從根本上解決了IoU所面臨的所有問題：

1. 平滑且有意義的梯度：無論預測點與真實點相距多遠，基於距離的損失函數總能提供一個平滑、連續且有意義的梯度信號。一個偏離5像素的預測會比偏離2像素的預測受到更大的懲罰，而模型也能清晰地知道需要朝哪個方向進行修正以減小損失。
2. 優化的「可學習性」：如果說IoU的損失地貌是充滿懸崖峭壁的險峻山谷，那麼距離的損失地貌就是一個平滑的碗狀盆地。在這樣的地貌中，優化器可以輕鬆地沿著梯度方向穩定下降，最終收斂到盆地的最低點，即亞像素級別的精確座標。這極大地提升了任務的「可學習性」（learnability）。
3. 目標與優化的統一：此方法完美地統一了任務的最終目標（最小化座標誤差）和模型的優化過程。模型在訓練中每一步的努力，都是在直接地、高效地朝著提升定位精度的方向前進。

綜上所述，從物件偵測的「邊界框」思維轉向關鍵點估計的「座標」思維，是一次解決問題的典範轉移。它不僅僅是技術選型上的差異，更是對問題本質進行更深刻理解後所採取的、在數學上更為優雅、在實踐中更為精確的解決路徑 [1]。

第二節：YOLOv8 姿態估計框架：架構與機制解析

為了將關鍵點估計的理論思想付諸實踐，我們需要一個強大且靈活的深度學習框架。Ultralytics開發的YOLOv8框架，憑藉其卓越的性能和高度模組化的設計，成為了理想的選擇。其內建的姿態估計（pose estimation）模式，雖然最初是為人體姿態分析而設計，但其底層架構完全可以被改用於我們的羽球場交點偵測任務 [2]。

2.1 架構概覽：一個雙頭並行的預測系統

YOLOv8-pose模型並非一個全新的架構，而是在成熟的YOLOv8物件偵測器基礎上進行的擴展 [8]。其核心架構由三個部分組成：骨幹網絡（Backbone）、頸部網絡（Neck）和預測頭（Head）。

1. 骨幹網絡 **(Backbone)**：與標準YOLOv8相同，通常採用基於CSPDarknet的設計，負責從輸入影像中提取一系列由淺到深、由粗到細的階層式特徵圖（feature maps）[9]。

2. 頸部網絡 **(Neck)**：採用如特徵金字塔網絡（FPN）和路徑聚合網絡（PANet）相結合的結構 [1]。其作用是融合來自骨幹網絡不同層級的特徵圖，將高層次的語義資訊（有助於分類）與低層次的空間細節資訊（有助於定位）結合起來，生成多個不同尺度的特徵圖，以應對不同大小的目標 [1]。
3. 預測頭 **(Head)**：這是YOLOv8-pose與標準YOLOv8最關鍵的區別所在。YOLOv8-pose採用了一個「雙頭」設計。在頸部網絡輸出的每一個尺度特徵圖上，都並行連接了兩個預測頭：
    - 偵測頭 **(Detect Head)**：與標準YOLOv8的偵測頭完全相同，負責預測物件的類別（class）和邊界框（bounding box）。
    - 姿態頭 **(Pose Head)**：這是專為姿態估計新增的預測頭，負責預測與每個偵測到的物件相關聯的關鍵點座標（keypoints）[9]。

這種並行設計意味著模型在一次前向傳播中，能夠同時完成「這是一個什麼類型的交點？」、「它大概在影像的哪個區域？」以及「它精確的幾何中心座標在哪裡？」這三個子任務。

2.2 端到端的預測流程

當一張包含羽球場的影像輸入到YOLOv8-pose模型後，其內部的端到端預測流程如下：

1. 特徵提取與融合：影像首先通過骨幹網絡和頸部網絡，生成一組包含豐富語義和空間資訊的多尺度特徵圖。
2. 網格化與錨點：與所有YOLO模型一樣，每個尺度的特徵圖被劃分成一個網格。網格中的每個單元（grid cell）負責預測其中心點落入的物件。
3. 並行預測：對於每個負責預測的網格單元，兩個預測頭同時工作：
    - 偵測頭輸出關於邊界框的預測，包括框的中心點座標 (x,y)、寬高 (w,h)，以及每個交點類別（L-junction, T-junction, cross-junction）的信賴度分數。
    - 姿態頭則輸出一個或多個關鍵點的預測。在我們的任務中，它將預測單一關鍵點相對於該網格單元的偏移座標。
4. 解碼與後處理：模型輸出的原始預測值是相對於網格和錨點的偏移量。這些值經過解碼過程，轉換為影像空間中的絕對座標。最後，應用非極大值抑制（Non-Max Suppression, NMS）演算法，去除對同一個交點的多個冗餘預測，保留信賴度最高的唯一結果 [2]。

最終，模型的輸出是一個結構化的列表。列表中的每一個元素都代表一個被偵測到的交點，並包含以下資訊：其類別標籤、一個用於界定其範圍和提供尺度的邊界框，以及最關鍵的——一個高精度的 (x,y) 關鍵點座標 [10]。這個流程將一個複雜的幾何分析問題，轉化為一個標準化的、端到端的深度學習任務。

第三節：數據集策劃與標註實踐指南

一個高品質、標註精確的數據集是訓練出高性能模型的基石。對於將交點偵測建構為關鍵點估計任務而言，數據的準備流程既有傳承於傳統物件偵測的部分，也包含了針對關鍵點任務的特定要求。

3.1 權威性標註協議

為了確保數據的一致性和高品質，必須制定並嚴格遵守一套標註協議。一個高效的標註流程，結合清晰的規則，是專案成功的先決條件。

1. 物件類別定義：首先，明確定義三個目標類別：
    - L-junction：由兩條相互垂直的線段構成的直角，通常位於場地邊角或發球線與邊線的交點。
    - T-junction：由一條線段的端點與另一條線段的中間部分相交構成，主要存在於單打後發球線與邊線的交點。
    - cross-junction：由兩條線段相互交叉構成，僅存在於場地中心的兩個中線與前發球線的交點。
2. 邊界框標註規則：儘管我們的最終目標是關鍵點，但標註一個緊密的邊界框仍然至關重要。邊界框應盡可能小，剛好包圍住構成交點的線條相交的核心區域 [1]。這個邊界框在後續的訓練和評估中扮演著提供物體尺度的關鍵角色。
3. 關鍵點放置規則：對於每一個標註的邊界框，必須在其內部放置一個單一的關鍵點。這個關鍵點的位置必須是交點的幾何中心，即線條理論上相交的那個數學點。標註員應盡可能放大影像，以達到亞像素級別的放置精度。
4. 標註工具選擇：雖然可以使用如LabelImg或LabelMe等傳統工具，但強烈推薦使用更現代化的、原生支持關鍵點標註的平台 [12]。Roboflow是一個優秀的選擇，它提供了專為關鍵點偵測專案設計的用戶界面，允許用戶先定義一個包含各類別及其對應關鍵點結構的「骨架」（skeleton），然後在標註時直接為每個邊界框添加對應的關鍵點 [13]。CVAT也是一個強大的選項，特別是其視訊標註功能，可以極大地提升從比賽錄影中提取和標註數據的效率 [1]。

3.2 精通YOLOv8-Pose標註格式

YOLOv8-pose使用一種特定的文本格式來存儲標註信息。每個影像對應一個 .txt 檔案，檔案中的每一行代表影像中的一個物件實例 [10]。理解並正確生成此格式是成功訓練模型的技術前提。
對於我們的羽球場交點偵測任務，每一行標註的格式如下：
<class-index> <x_center> <y_center> <width> <height> <px1> <py1> <v1>
下面以一個具體的例子來解析這個格式。假設我們在一個1920x1080的影像中標註一個T-junction，其類別ID為1。

- 邊界框：其中心點在 (960, 540)，寬高為 (50, 50)。
- 關鍵點：其精確的幾何中心也在 (960, 540)，且清晰可見。

該標註行將會是：
1 0.5000 0.5000 0.0260 0.0463 0.5000 0.5000 2
各個欄位的詳細解釋如下：

- <class-index>：物件的類別索引，從0開始。假設L-junction為0，T-junction為1，cross-junction為2，此處為1。
- <x_center> <y_center>：邊界框中心點的歸一化座標。計算方式為 x_pixel / image_width 和 y_pixel / image_height。此處為 960 / 1920 = 0.5 和 540 / 1080 = 0.5。

- <width> <height>：邊界框的歸一化寬高。計算方式為 box_width / image_width 和 box_height / image_height。此處為 50 / 1920 ≈ 0.0260 和 50 / 1080 ≈ 0.0463。
- <px1> <py1>：第**1**個關鍵點的歸一化座標。計算方式與邊界框中心點相同。此處為 960 / 1920 = 0.5 和 540 / 1080 = 0.5。
- <v1>：第**1**個關鍵點的可見性標誌（visibility flag）。

這個格式清晰地展示了邊界框和關鍵點在YOLOv8-pose框架中的共生關係。

3.3 善用可見性標誌以提升模型穩健性

可見性標誌 v 是YOLOv8-pose標註格式中一個極其強大但常被忽視的特性。它允許我們向模型傳達關於關鍵點狀態的額外信息 [10]。其標準定義如下：

- v=0：未標註的關鍵點（在我們的任務中不應使用）。
- v=1：已標註但被遮擋（不可見）的關鍵點。
- v=2：已標註且清晰可見的關鍵點。

在羽球比賽的真實場景中，交點被球員的腳、球拍或羽球短暫遮擋的情況非常普遍 [1]。正確使用可見性標誌對於訓練一個能在複雜場景下穩定工作的模型至關重要。我們的標註協議應包含以下規則：

- 對於清晰可見的交點：標註其邊界框和關鍵點，並將可見性標誌設為 v=2。
- 對於部分或完全被遮擋的交點：如果標註員仍能根據上下文（如可見的部分線條）準確推斷出交點的確切位置，則應標註其邊界框和推斷出的關鍵點位置，並將可見性標誌設為 v=1。
- 對於完全無法確定位置的交點：則不應進行標註，以避免向模型引入模糊或錯誤的信息。

通過使用 v=1 標誌，我們實際上是在明確地指導模型學習應對遮擋。在計算損失時，對於可見性為1的關鍵點，其定位損失會被忽略或給予極低的權重 [14]。這避免了模型因無法預測一個它根本看不見的點而受到不公平的懲罰，從而穩定訓練過程。更重要的是，模型仍然會學習該位置存在一個特定類別的物件（來自邊界框和分類損失）。這會激勵模型學習從周圍的上下文特徵（例如「一條白線延伸到一隻運動鞋底下」）來推斷被遮擋交點的存在。這種能力是模型在真實比賽視頻中實現高召回率和低漏檢率的關鍵。

表 **3.1**：標註格式對比：標準物件偵測 **vs.** 關鍵點估計

特徵	標準YOLO物件偵測格式	YOLOv8-Pose 關鍵點估計格式 (本任務適用)
目標	定位並分類一個有面積的「物件」。	定位並分類一個「物件」，並找出其內部的精確「點」。
格式結構	<class> <x> <y> <w> <h>	<class> <x> <y> <w> <h> <px> <py> <v>
座標部分	4個值：邊界框的中心點、寬、高。	7個值：邊界框的4個值，加上關鍵點的2個座標和1個可見性標誌。
核心資訊	邊界框的位置和大小。	關鍵點的精確座標。
示例	1 0.5 0.5 0.026 0.046	1 0.5 0.5 0.026 0.046 0.5 0.5 2

第四節：模型配置、訓練與微調

將精心準備的數據集轉化為一個高性能的模型，需要對訓練框架進行精確的配置，並採用合理的訓練策略。本節將詳細闡述如何配置YOLOv8-pose框架，執行訓練流程，並深入探討其中關鍵組件的 symbiotic role。

4.1 為關鍵點訓練配置 data.yaml 文件

data.yaml 文件是連接數據集與YOLOv8訓練框架的橋樑，它定義了數據集的路徑、類別信息以及最關鍵的關鍵點結構 [10]。一個錯誤的配置將直接導致訓練失敗。對於我們的羽球場交點偵測任務，一個完整且正確的
badminton_intersections.yaml 文件應如表4.1所示。

表 **4.1**：**badminton_intersections.yaml** 配置文件詳解

YAML

```
# 數據集根目錄路徑
path:../datasets/badminton_intersections

# 訓練集、驗證集、測試集影像的路徑 (相對於 'path')
train: 'images/train'
val: 'images/val'
test: 'images/test' # 可選，用於最終評估

# 類別定義
nc: 3  # number of classes, 類別總數
names:
  - L-junction
  - T-junction
  - cross-junction

# 關鍵點結構定義 (至關重要)
kpt_shape:   # [number of keypoints, number of dimensions]
flip_idx:      # 水平翻轉時關鍵點的索引交換列表
```

對此配置文件中的關鍵參數進行深入解析：

- path, train, val, nc, names：這些參數與標準物件偵測的配置相同，分別定義了數據路徑和

類別信息 [15]。

- kpt_shape: :這是整個配置文件中最核心的部分，專為姿態估計任務設計 [10]。
    - 第一個數字 1 代表每個物件實例所擁有的關鍵點數量。在我們的任務中，每個交點（無論是L、T還是Cross型）都只由一個中心的幾何點來定義，因此這個值為1。
    - 第二個數字 3 代表每個關鍵點的維度。我們不僅需要預測其二維座標 (x,y)，還需要預測其可見性 v。因此，維度為3。如果任務不需要處理遮擋，可以設為2，只預測 (x,y)。
- flip_idx: :這個參數定義了在使用水平翻轉（horizontal flip）數據增強時，關鍵點索引應該如何交換 [10]。例如，在人體姿態估計中，左眼和右眼的索引需要在翻轉後互換。由於我們的任務中每個物件只有一個中心關鍵點，它自身就是對稱的，不存在需要交換的配對，因此列表只包含其自身的索引 ``。

4.2 邊界框在關鍵點估計中的共生作用

一個常見的困惑是：「既然基於IoU的損失函數不適用於點狀特徵，為何我們仍需費力標註並讓模型預測邊界框？」這個問題的答案揭示了YOLOv8-pose架構設計的精妙之處。邊界框在此任務中扮演著兩個不可或缺的共生角色。

4.2.1 角色一：作為尺度歸一化器

邊界框的首要且最關鍵的作用，是為關鍵點定位的評估和優化提供一個動態的、與物體相關的尺度參照。如前文所述，關鍵點定位的黃金標準是物件關鍵點相似度（OKS），其核心公式為 $OKS=\exp(-d2/2s2k2)$ [3]。

在這個公式中，d 是預測點與真實點之間的歐幾里得距離。如果沒有歸一化，一個5像素的預測誤差對於遠景中只有10x10像素的交點來說是致命的，但對於近景特寫中100x100像素的交點來說則可能是個微不足道的誤差。變數 s（物體尺度）正是為了解決這個問題而引入的。s2 通常被定義為物體真實的分割掩碼面積。在實踐中，由於分割掩碼標註成本高昂，通常使用邊界框的面積（即 width * height）作為 s2 的一個高效且合理的近似 [14]。

因此，通過在OKS計算中用邊界框面積來歸一化距離 d，損失函數變得具有尺度不變性。模型受到的懲罰不再是絕對的像素誤差，而是相對於交點自身大小的相對誤差。這使得模型能夠在包含各種遠近、縮放和視角變換的複雜場景中，以一種公平且一致的方式學習精確定位。邊界框的存在，是實現這種高級、自適應優化機制的基礎。

4.2.2 角色二：作為隱式注意力機制

邊界框的第二個作用，源於YOLOv8-pose的架構設計本身。模型並非直接在整張圖上暴力搜索所有可能的關鍵點，而是採用了一種更為高效的、分而治之的策略。其內在邏輯可以看作一個兩階段的注意力過程：

1. 粗粒度定位（**Coarse Localization**）：模型的偵測頭首先回答一個問題：「影像的哪些區域可

能包含一個交點？」它通過預測邊界框來完成這個任務，將注意力從無關的背景（如觀眾席、球員身體）轉移到包含場地線的候選區域。

2. 細粒度定位（**Fine-grained Localization**）：一旦一個候選區域（即邊界框）被確定，姿態頭就在這個高度相關的局部區域內，集中精力解決下一個問題：「在這個小區域內，精確的交點中心在哪裡？」

這種「先找框，再找點」的流程，實際上是將一個複雜的全局搜索問題分解為多個簡單的局部搜索問題。邊界框在這裡充當了一個隱式的注意力機制，它為後續的精確關鍵點回歸提供了必要的局部上下文，極大地降低了問題的複雜度，提高了模型的學習效率和最終的定位精度。

4.3 訓練執行與超參數策略

配置完成後，即可啟動訓練。訓練可以通過命令列介面（CLI）或Python API兩種方式執行 [17]。

命令列介面（**CLI**）執行範例：

Bash

```
yolo task=pose mode=train model=yolov8n-pose.pt
data=./config/badminton_intersections.yaml imgsz=1280 epochs=100 batch=8
name=badminton_kpts_v1
```

Python API 執行範例：

Python

```
from ultralytics import YOLO

# 載入一個預訓練的姿態估計模型
model = YOLO('yolov8n-pose.pt')

# 開始訓練
results = model.train(
   task='pose',
   data='./config/badminton_intersections.yaml',
   imgsz=1280,
   epochs=100,
   batch=8,
   name='badminton_kpts_v1'
)
```

對關鍵超參數的選擇策略進行說明：

- task=pose：這是必須顯式聲明的參數，用以告知YOLOv8框架啟用姿態估計模式，加載對應的Pose預測頭並使用關鍵點損失函數。
- model=yolov8n-pose.pt：強烈建議從一個在大型數據集（如COCO-Pose）上預訓練過的姿態模型開始 [6]。這個預訓練模型已經學會了通用的低層特徵（如邊緣、角點）和關鍵點定位的基本能力。在此基礎上進行微調（fine-tuning），將極大地加速收斂過程，並通常能達到比從零開始訓練（from scratch）更高的最終精度。
- imgsz=1280：對於包含大量微小目標的任務，使用高解析度輸入是提升效能的關鍵權衡 [1]。較低的解析度（如默認的640）會在圖像縮放過程中丟失遠景交點的關鍵像素細節，導致模型無法學習。儘管這會增加計算成本和顯存消耗，但對於追求高精度而言是必要的。
- batch=8：批次大小主要受限於GPU顯存。在顯存允許的範圍內，較大的批次通常能提供更穩定的梯度，有助於訓練。8是一個在多數中高階GPU上（如12GB+ VRAM）運行1280解析度訓練時較為可行的值。

通過上述配置和命令，我們便可以啟動一個專為高精度羽球場交點偵測而設計的、基於關鍵點估計的YOLOv8模型訓練流程。

第五節：核心機制：解構關鍵點損失函數

要深刻理解模型如何學習精確定位，就必須深入其數學核心——損失函數（Loss Function）。YOLOv8-pose的總損失是一個由多個部分加權構成的複合損失，它協同工作，共同引導模型向著期望的目標優化。

5.1 複合損失函數的構成

YOLOv8-pose的總損失 Ltotal 可以表示為三個主要分量的加權和 [1]：

$$L_{total} = w_{cls} \cdot L_{cls} + w_{box} \cdot L_{box} + w_{kpts} \cdot L_{kpts}$$

其中，w_{cls}, w_{box}, w_{kpts} 是用於平衡各個損失分量重要性的權重係數。

- 分類損失 **Lcls**：這部分負責懲罰模型對交點類別（L、T、Cross）的錯誤分類。YOLOv8通常採用二元交叉熵損失（BCEWithLogitsLoss），它將多類別分類問題視為多個獨立的二元分類問題，具有良好的靈活性 [1]。
- 邊界框損失 **Lbox**：這部分負責優化邊界框的定位。YOLOv8採用了如完整交並比損失（Complete IoU, CIoU）或分佈焦點損失（Distribution Focal Loss, DFL）的組合 [1]。如前文所述，在本任務中，
  Lbox 的主要作用是驅使模型預測出一個能夠準確反映交點尺度的邊界框，為關鍵點損失的計算提供可靠的尺度歸一化因子 s。
- 關鍵點損失 **Lkpts**：這是我們任務的核心，專門負責懲罰關鍵點座標的預測誤差。YOLOv8-pose的關鍵點損失是基於物件關鍵點相似度（OKS）來計算的。

5.2 物件關鍵點相似度（OKS）損失的深度解析

關鍵點損失 Lkpts 通常被定義為 1−OKS，即最大化OKS分數 [19]。讓我們再次深入審視OKS的計算公式，並剖析其每個組件的意義 [3]：

$$OKS=\Sigma i\delta(vi>0)\Sigma i\exp(-2s2ki2di2)\delta(vi>0)$$

- i：表示物體的第 i 個關鍵點。在我們的任務中，由於每個交點只有一個關鍵點，所以 i 總為1。
- di：預測的第 i 個關鍵點與真實（ground truth）的第 i 個關鍵點之間的歐幾里得距離。這是公式的核心，直接度量了定位的像素級誤差。
- s：物體尺度（object scale），通常由真實邊界框的面積開根號得到，即 s=wgt·hgt。s2 項的存在使得 di2 被物體自身的面積所歸一化，從而實現了尺度不變性。
- ki：一個預先定義的、針對第 i 類關鍵點的常數。這個常數反映了該類關鍵點的標註難度或本身的可變性，通常由對大量數據進行標註實驗，統計人類標註員的標準差（standard deviation）而來。例如，在人體姿態中，標註腳踝的標準差會比標註鼻尖要大，因此腳踝的 k 值會更大，允許更大的預測容忍度。
- exp(·)：指數函數將一個範圍為 [0,∞) 的歸一化平方距離，優雅地映射到一個範圍為 (0,1] 的相似度分數。當距離為0時，OKS為1；當距離趨於無窮時，OKS趨於0。這種形式完美地模擬了IoU的行為。
- δ(vi>0)：狄拉克δ函數，用於處理可見性。它確保只有被標註的關鍵點（即 vi=1 或 vi=2）才會被納入OKS的計算中。這意味著對於被遮擋（vi=1）的點，雖然我們標註了它，但在計算損失時，模型不會因為預測不準而受到懲罰。

5.3 關鍵點常數 k 的處理策略：從實用到前沿

在OKS公式中，唯一懸而未決的變數是 ki。對於COCO這樣成熟的數據集，其17個人體關鍵點的 k 值是公開的、經過研究者精心標定的標準值 [4]。然而，對於我們自定義的「羽球場交點」這個新任務，並不存在現成的
k 值。這是一個展現工程智慧和研究深度的地方。

策略一（實用主義方法）：假設 **k** 值為常數

在缺乏先驗知識的情況下，最直接和務實的方法是做出一個合理的簡化假設。我們可以假設三種交點（L、T、Cross）的幾何定義都非常清晰，因此人類標註員在標註它們時的難度和方差是基本一致的。

基於這個假設，我們可以為所有關鍵點設置一個統一的 k 值。由於 k 在公式中主要起到相對權衡的作用，將所有 ki 設為同一個常數（例如，簡單地設為1.0）在數學上是等效的。這意味著我們認為所有類型的交點在定位精度上的要求是相同的。大多數深度學習框架在處理自定義關鍵點任務時，如果用戶不特別指定，也通常會採用這種默認設置。對於快速搭建一個高性能的基線模型，這是一個完全可以接受且有效的策略。

策略二（研究導向方法）：經驗性標定自定義 **k** 值

為了追求極致的性能和理論上的完備性，我們可以借鑒COCO數據集標定 k 值的方法，為我們的任務進行一次小規模的標註實驗。這個過程本身就是一項有價值的小型研究，其步驟如下：

1. 數據抽樣：從我們的數據集中隨機抽取一小部分具有代表性的影像，例如100到200張，確保涵蓋不同類型、不同視角和不同清晰度的交點。

2. 多重標註：邀請多位（例如3-5位）標註員獨立地對這批抽樣影像進行關鍵點標註。
3. 統計方差：對於每一個交點實例，我們現在擁有了多個獨立的標註座標。計算這些座標點相對於其均值點的標準差（standard deviation）。
4. 計算 **k** 值：將所有L-junction實例的標準差進行平均，得到 kL−junction；同樣地，計算出 kT−junction 和 kcross−junction。這些經驗性測量出的標準差，就成為了我們任務專屬的、數據驅動的 k 值。

將這些經過經驗性標定的 k 值應用到OKS損失函數中，將使得我們的優化目標更加精確地反映羽球場交點這一特定領域的內在屬性。例如，如果實驗發現cross-junction由於線條更密集而更容易精確標註（標準差更小），那麼它的 k 值就會更小，模型在訓練時就會被驅使以更高的精度來定位cross-junction。

總結而言，YOLOv8-pose的損失函數是一個精心設計的系統。它通過邊界框損失來確保尺度感知的準確性，通過分類損失來保證類別識別的正確性，並通過基於OKS的關鍵點損失，以一種尺度不變且對遮擋魯棒的方式，專注於實現亞像素級的精確座標定位。

第六節：嚴謹的性能評估框架

評估模型的性能不僅是為了得到一個分數，更是為了深入理解模型的優勢與不足。對於關鍵點估計任務，採用正確的評估指標至關重要。使用不恰當的指標（如基於IoU的mAP）不僅會產生誤導性結論，也無法與學術界和工業界的標準實踐進行比較。

6.1 超越IoU：關鍵點精度的專用評估指標

我們必須明確，對於此任務，任何基於IoU的評估指標都是不適用且應被拋棄的。評估的重點應該是座標的接近程度，而非區域的重疊程度。為此，我們將採用兩種互補的、業界公認的標準指標：基於OKS的平均精確率均值（mAP）和百分比正確關鍵點（PCK）。

6.2 基於OKS的平均精確率均值（mAP）

這是姿態估計領域最權威、最全面的評估指標，也是COCO等標準競賽所採用的官方指標[5]。它不僅評估定位精度，還綜合考量了分類的正確性和模型預測的信賴度。其計算流程嚴謹而複雜：

1. 計算**OKS**：對於模型輸出的每一個預測（包含類別、邊界框、關鍵點和信賴度分數），計算它與數據集中所有同類別的真實物體之間的OKS分數。
2. 匹配與分類：設定一個OKS閾值（例如，OKS_threshold = 0.5）。對於一個預測，如果它與某個尚未被匹配的真實物體之間的OKS分數大於該閾值，則此預測被視為一個真正例（**True Positive, TP**）。如果其OKS分數低於閾值，或者它匹配到了一個已經被其他更高信賴度預測匹配過的真實物體，則它被視為一個假正例（**False Positive, FP**）。數據集中所有沒有被任何TP預測匹配到的真實物體，則被視為假反例（**False Negative, FN**）。
3. 繪製**P-R**曲線：模型的每個預測都有一個信賴度分數。通過從高到低遍歷所有可能的信賴度分數閾值，在每個閾值點上，我們都可以計算出當前的精確率（Precision = TP / (TP + FP)）

和召回率（Recall = TP / (TP + FN)）。將這些（Recall, Precision）對繪製成圖，就得到了該類別在特定OKS閾值下的精確率-召回率曲線（**Precision-Recall Curve**）。

4. 計算**AP**：該P-R曲線下的面積（Area Under the Curve, AUC）就是該類別在該OKS閾值下的平均精確率（**Average Precision, AP**）。例如，AP@0.5 指的就是在OKS閾值為0.5時計算出的AP值。
5. 計算**mAP**：為了得到一個不受特定閾值選擇影響的綜合指標，COCO標準做法是計算一系列OKS閾值（從0.50到0.95，步長為0.05）下的AP值，然後將它們平均起來，得到最終的AP@[.50:.05:.95]。最後，將數據集中所有類別（L、T、Cross）的該AP值再做一次平均，就得到了最終的評估指標——平均精確率均值（**mean Average Precision, mAP**）。

實踐中的注意事項：
一個需要特別注意的細節是，Ultralytics框架在訓練過程中實時顯示的驗證mAP，為了計算效率，可能採用的是一種簡化的、基於距離的度量，而非嚴格的OKS 23。雖然這對於監控訓練趨勢非常有用，但為了獲得可供發表或進行嚴謹比較的最終結果，標準做法是：將模型的預測結果導出為COCO標準的JSON格式，然後使用官方的
pycocotools評估腳本來進行計算 [5]。這確保了評估結果的公正性和可複現性。

6.3 百分比正確關鍵點（PCK）

雖然mAP是全面的，但其計算過程複雜，數值不夠直觀。為此，我們可以引入一個更易於理解的輔助指標——百分比正確關鍵點（Percentage of Correct Keypoints, PCK）[24]。
PCK的定義非常直接：如果一個預測的關鍵點與其對應的真實關鍵點之間的距離，小於某個預設的閾值，那麼這個關鍵點就被認為是「正確的」。PCK就是數據集中所有被正確檢測的關鍵點所佔的百分比。
關鍵在於閾值的設定。為了消除物體尺度的影響，這個距離閾值通常是相對於物體自身的大小來定義的。對於我們的任務，一個合理的PCK定義是：
如果預測點與真實點之間的歐幾里得距離小於 α×max(wbbox,hbbox)，則該點為正確。
其中，wbbox 和 hbbox 是該交點的真實邊界框的寬和高，α 是一個預設的容忍度係數（例如0.1, 0.2等）。
我們可以報告不同容忍度下的PCK值，例如PCK@0.1（要求誤差在邊界框尺寸的10%以內） 和PCK@0.2。PCK提供了一個非常直觀的性能快照，例如「我們的模型能夠以小於其邊界框尺寸20%的誤差，精確定位98%的場地交點」，這對於向非技術背景的相關方解釋模型性能非常有幫助。

表 **6.1**：關鍵點估計任務評估指標對比分析

指標	衡量維度	對尺度敏感性	對點狀特徵適用性	可解釋性
IoU-based mAP	面積重疊度	極度敏感	極差。微小偏移導致分數劇變，無法有效評估定位精度。	低。IoU值與實際像素誤差的關係非線性。
OKS-based mAP	綜合性能（定位、分類、信賴度）	尺度不變。通過邊界框面積s進行歸	極佳。專為關鍵點設計，是學術和工	中。單一mAP值綜合了多方面信息，

		一化。	業界的黃金標準。	但背後機制複雜。
PCK	定位精度	尺度不變。通過物體尺寸（如邊界框）進行歸一化。	極佳。直接衡量歸一化後的距離誤差，非常適合定位任務。	高。結果易於理解，如「95%的點在閾值內」。

第七節：綜合結論、建議與未來展望

本報告系統性地闡述了採用關鍵點估計方法，特別是利用YOLOv8-pose框架，來解決羽球場地幾何交點偵測問題的完整方法論。從理論基礎的典範轉移，到數據準備、模型配置、損失函數機制及性能評估，我們構建了一個端到端的、旨在實現高精度定位的技術框架。

7.1 端到端實施清單

為了便於實踐者快速上手，現將整個流程總結為一個清晰的實施清單：

1. 確立思維：摒棄傳統物件偵測的「框」思維，擁抱關鍵點估計的「點」思維，認識到後者在幾何定位任務上的根本優勢。
2. 數據標註：
    - 使用支持關鍵點標註的工具（如Roboflow, CVAT）。
    - 為每個交點標註一個緊密的邊界框和一個精確的中心關鍵點。
    - 嚴格遵循YOLOv8-pose的 .txt 標註格式：<class> <x_c> <y_c> <w> <h> <px> <py> <v>。
    - 善用可見性標誌 v：可見點設為2，被遮擋但可推斷的點設為1。
3. 環境配置：
    - 創建 data.yaml 配置文件。
    - 確保 nc 和 names 與類別定義一致。
    - 關鍵配置：設置 kpt_shape: 和 flip_idx: 。
4. 模型訓練：
    - 使用 task=pose 參數啟動訓練。
    - 從預訓練的 yolov8n-pose.pt 或更大尺寸的模型開始微調。
    - 採用高解析度輸入，例如 imgsz=1280，以保留微小目標的細節。
5. 性能評估：
    - 主要評估指標採用基於**OKS**的**mAP**（特別是mAP@0.5:0.95），以實現與學術界的對標。
    - 輔助評估指標採用**PCK**（如PCK@0.1），以提供直觀的精度解釋。
    - 避免使用任何基於IoU的指標進行最終評估。

7.2 預期性能增益與應用前景

與傳統的、基於邊界框的物件偵測方法相比，採用本報告提出的關鍵點估計方法，預期將在定位精度上取得顯著的、量級上的提升。具體而言，我們預計在要求嚴格的 mAP@0.5:0.95 指標上，關鍵點模型的表現將遠超物件偵測模型。這種精度的飛躍，不僅僅是數字上的提升，它直接賦能了一系列過去難以實現的下游高級應用。
一個高精度的交點偵測器，本身就是一個強大的工具，可用於輔助裁判系統判斷落點和發球違例。但其更大的價值在於，它是一項關鍵的使能技術（enabling technology），為實現更宏大的體育分析系統奠定了基礎。

7.3 未來方向：實現全自動化的場地透視變換

羽球比賽的轉播視角多變，這給所有基於位置的量化分析帶來了巨大挑戰。一個終極的目標是能夠將任何視角的比賽影像，自動「拉平」到一個標準的、統一的2D俯視圖上。這個過程被稱為透視變換或單應性變換（**Homography**）[1]。
要計算出這個變換所需的單應性矩陣（homography matrix），至少需要四對對應點——即四個在源影像（攝像機視角）和目標影像（標準2D場地圖）中都已知位置的點。羽球場上的交點，其在真實世界中的相對位置是固定的、已知的（可從BWF官方規則查得）[1]，因此它們是理想的特徵點。
這正是本報告所開發的高精度關鍵點偵測器發揮其終極價值的所在：

1. 提供源點：我們的模型能夠從任意視角的影像中，穩定、精確地偵測出多個交點的 (x,y) 座標。這些就是計算單應性所需的「源點」。
2. 提供目標點：根據官方場地尺寸，我們可以預先繪製一張標準的2D俯視圖，圖上所有交點的座標都是已知的「目標點」。
3. 計算變換：利用OpenCV等視覺庫，只需將至少四對「源點」和「目標點」作為輸入，即可輕鬆計算出準確的單應性矩陣。

一個定位精度不高的偵測器，其輸出的源點會帶有抖動和誤差，導致計算出的單應性變換極不穩定，使得拉平後的影像產生扭曲和「搖晃」。而我們通過關鍵點估計方法所追求的亞像素級精度，正是為了確保源點的穩定性，從而生成一個穩定、精確、可靠的單應性變換。
一旦實現了這種全自動的透視變換，整個體育分析的潛力將被極大釋放。分析師將能夠在一個統一的座標系下，量化分析球員的跑動覆蓋、回球落點分佈、戰術模式等，而無需再受制於多變的攝影機角度。這不僅將徹底改變專業球隊的技戰術分析方式，也將為電視轉播和觀眾體驗帶來革命性的創新。因此，本報告所構建的高精度交點偵測器，不僅是解決了一個具體的視覺任務，更是通往下一代智慧體育分析系統的關鍵一步。

引用的著作

1. 羽球場交點YOLO偵測與分類_.pdf
2. YoloV8 Pose Estimation Tutorial - Dev-kit, 檢索日期：7月 31, 2025, https://dev-kit.io/blog/machine-learning/yolov8-pose-estimation
3. Human Pose Estimation 2023 Guide | SoftwareMill, 檢索日期：7月 31, 2025, https://softwaremill.com/human-pose-estimation-2023-guide/
4. Keypoint Similarity (ks). The ks between two detections, eye (red) and... - ResearchGate, 檢索日期：7月 31, 2025, https://www.researchgate.net/figure/Keypoint-Similarity-ks-The-ks-between-two

-detections-eye-red-and-wrist-green-and_fig1_323793190
5. Keypoint estimation - SVIRO, 検索日期：7月 31, 2025, https://sviro.kl.dfki.de/keypoint-estimation/
6. Pose Estimation - Ultralytics YOLO Docs, 検索日期：7月 31, 2025, https://docs.ultralytics.com/tasks/pose/
7. Pose Estimation with Ultralytics YOLOv8, 検索日期：7月 31, 2025, https://www.ultralytics.com/blog/pose-estimation-with-ultralytics-yolov8
8. YOLOv8 Pose: The pose estimation that transforms computer vision - OpenSistemas, 検索日期：7月 31, 2025, https://opensistemas.com/en/yolov8-pose-transforming-pose-estimation/
9. Enhanced human pose estimation using YOLOv8 with Integrated SimDLKA attention mechanism and DCIOU loss function: Analysis of human body behavior and posture | PLOS One, 検索日期：7月 31, 2025, https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0318578
10. Pose Estimation Datasets Overview - Ultralytics YOLO Docs, 検索日期：7月 31, 2025, https://docs.ultralytics.com/datasets/pose/
11. YOLOv8-Pose annotations format · Issue #1970 - GitHub, 検索日期：7月 31, 2025, https://github.com/ultralytics/ultralytics/issues/1970
12. Pose Estimation Custom Dataset · Issue #2117 - GitHub, 検索日期：7月 31, 2025, https://github.com/ultralytics/ultralytics/issues/2117
13. How to Train a Custom YOLOv8 Pose Estimation Model - Roboflow Blog, 検索日期：7月 31, 2025, https://blog.roboflow.com/train-a-custom-yolov8-pose-estimation-model/
14. Object Keypoint Similarity in Keypoint Detection - LearnOpenCV, 検索日期：7月 31, 2025, https://learnopencv.com/object-keypoint-similarity/
15. YOLOv8 Setup Tutorial for Object Detection | Exxact Blog, 検索日期：7月 31, 2025, https://www.exxactcorp.com/blog/deep-learning/yolov8-setup-tutorial-for-object-detection
16. 'yolov8n-pose.yaml' · Issue #721 · ultralytics/hub - GitHub, 検索日期：7月 31, 2025, https://github.com/ultralytics/hub/issues/721
17. Model Training with Ultralytics YOLO, 検索日期：7月 31, 2025, https://docs.ultralytics.com/modes/train/
18. Enhanced human pose estimation using YOLOv8 with Integrated SimDLKA attention mechanism and DCIOU loss function: Analysis of human body behavior and posture - ResearchGate, 検索日期：7月 31, 2025, https://www.researchgate.net/publication/391525914_Enhanced_human_pose_estimation_using_YOLOv8_with_Integrated_SimDLKA_attention_mechanism_and_DCIOU_loss_function_Analysis_of_human_body_behavior_and_posture
19. ProbPose: A Probabilistic Approach to 2D Human Pose Estimation - arXiv, 検索日期：7月 31, 2025, https://arxiv.org/html/2412.02254v1
20. Pose Estimation. Metrics.. Fight detection, sport exercise... | by Aliaksandr Stasiuk - Medium, 検索日期：7月 31, 2025, https://stasiuk.medium.com/pose-estimation-metrics-844c07ba0a78
21. How to calculate object keypoint similarity - Stack Overflow, 検索日期：7月 31, 2025,

https://stackoverflow.com/questions/68250191/how-to-calculate-object-keypoint-similarity
22. On the Calibration of Human Pose Estimation - arXiv, 検索日期：7月 31, 2025, https://arxiv.org/html/2311.17105v1
23. Yolov8-pose mAP50 meaning ? · Issue #7334 - GitHub, 検索日期：7月 31, 2025, https://github.com/ultralytics/ultralytics/issues/7334
24. mmpose/mmpose/evaluation/metrics/keypoint_2d_metrics.py at main · open-mmlab/mmpose - GitHub, 検索日期：7月 31, 2025, https://github.com/open-mmlab/mmpose/blob/main/mmpose/evaluation/metrics/keypoint_2d_metrics.py
25. Percentage of correct keypoints (PCK) - Metrics - OECD.AI, 検索日期：7月 31, 2025, https://oecd.ai/en/catalogue/metrics/percentage-of-correct-keypoints-pck
26. cbsudux/Human-Pose-Estimation-101 - GitHub, 検索日期：7月 31, 2025, https://github.com/cbsudux/Human-Pose-Estimation-101