VICE: View-Invariant Chess Estimation

Kevin Zhu[a], Alexander Wong and John McPhee[b]
University of Waterloo, Canada

Keywords: Chess, Automated Digitization, Computer Vision, Deep Learning.

Abstract: A digitized chess match offers chess players a convenient way to study previous matches. However, manually recording a large number of matches can be laborious, while automated methods are usually hardware-based, requiring expensive chessboards. Computer vision provides a more accessible way to track matches from videos. However, current vision-based digitizers are often evaluated on images captured by cameras placed directly above a chessboard, and performance suffers when the camera angle is lower, limiting their applicability. Motivated to develop a more practical solution, we introduce VICE, a view-invariant chess estimator to digitize matches from camera angles not seen during training. Due to its small model size and computational efficiency, VICE is suitable for mobile deployment. By rearranging the framework for chess detection and incorporating prior information from chess and basic geometry, we simplify the chess estimation problem and mitigate the challenges that current chess digitizers struggle with, such as occlusion. We combine the board localization and chess piece detection phases of classical two-step chess estimation to develop a prototype for the first single-step chess digitizer. We show that, with minimal training data, our prototype can infer moves from camera angles that current chess digitizers cannot, while being much smaller in size.

1 INTRODUCTION

The current popularity growth in chess may rival the boom in 1972 (Keener, 2022), when the World Chess Championship match took place between American Bobby Fischer and Soviet Boris Spassky, at the height of the Cold War. Although the COVID-19 pandemic contributed to the current surge, with active users on *Chess.com* more than doubling from 8 million to 17 million from 2020 to 2022, it does not mean that people have permanently switched to competing online. In 2020, after the reopening of the Marshall Chess Club, one of the oldest chess clubs in the United States, the number of active members reached a record-high in the club's 107-year history.

Just like in any other sport, post-game analysis is important for a player's growth and development. A digitized chess match (Fig. 1) offers a convenient way to study both ongoing and previous matches. It is visual, compact, and can be uploaded to chess engines for analysis. However, manually recording in-person matches can be a tedious process. Current automated tracking methods are usually hardware-based (SquareOff, 2018; DGT, 1998) and require expensive boards, which could be inaccessible for many.

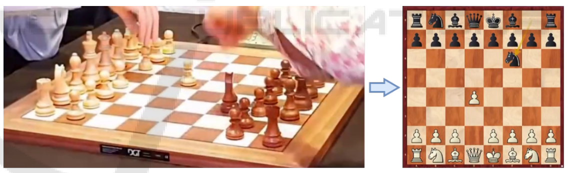


Figure 1: Digitized game state (right image) after move Nf6 of Ding vs. Aronian in the 2019 Tata Steel Blitz (ChessBaseIndia, 2020a). Ding (left player) starts his move immediately after Nf6, while Aronian's arm is still in frame, fully occluding some chess pieces around the A8 corner. At the current game state, there are no frames where human hands don't occlude part of the board.

Recent computer vision methods, such as LiveChess2FEN (Mallasén Quintana et al., 2021) and Chesscog (Wölflein and Arandjelović, 2021), have shown promising results on predicting a chess game state from an image. However, these methods are often evaluated on images taken from a bird's eye view (BEV), and performance suffers when the camera angle is lower. This limits the practicability of these methods in real-world scenarios. A scan on YouTube will show that most broadcast videos of chess are taken at a lower camera angle. At an amateur tournament or when partaking in a street chess match, it

[a] https://orcid.org/0000-0003-1585-6170
[b] https://orcid.org/0000-0003-3908-9519

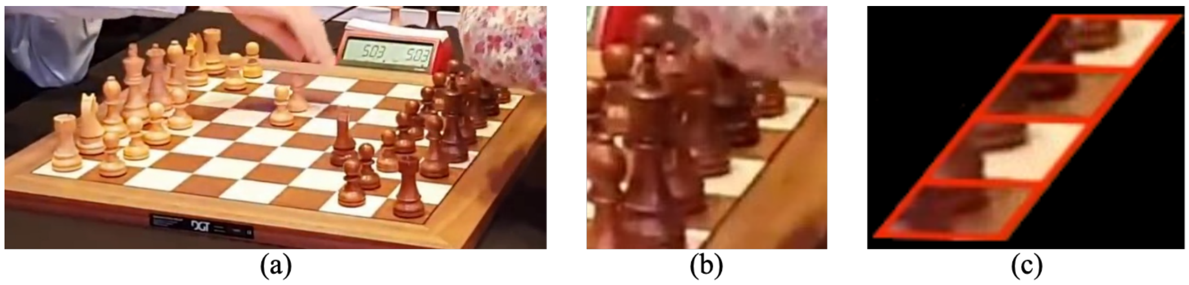


Figure 2: (a): An image corresponding to move c4. (b): Zooming in to the top right corner of (a), occlusion among chess pieces, occlusion from Aronian's arm and shadows make chess piece classification challenging. (c): The pixels within each square may not provide sufficient information to classify chess pieces since some chess sets have the same base for different chess pieces.

is also unlikely to have a personal phone or camera placed directly above the chessboard.

These vision-based methods first localize the chessboard using a combination of deep learning and traditional computer vision methods such as edge and corner detectors (Lu et al., 2015; Harris and Stephens, 1988), then classify the chess piece in each square of the chessboard using convolutional neural networks (CNNs). The board localization step implements many rule-based modules with hand-picked parameters to compute the geometry of the board from the detected edges and corners in the image. If problems occur during edge and corner estimation, errors could arise during the geometry calculations, terminating the entire process. This could happen when a human hand partially obstructs the board (Fig. 2a), a case that previous methods did not consider. The accuracy in the classification step suffers mainly due to occlusion among pieces (Fig. 2b). Even when occlusion is not the case, the pixels in each square sometimes don't provide enough information — at a low camera angle, only the base of the chess piece is visible in each square, and some chess sets have the same base for all pieces (Fig. 2c).

Motivated to develop a more robust method, we propose a view-invariant chess estimator (VICE) that is able to predict chess moves from unseen camera angles in a single step. We refer to camera angles and matches that do not appear in the training dataset as "unseen" camera angles and matches. The approach of current chess digitization methods is very similar to two-stage object detectors — a network generates region proposals in an image, then another network predicts the class of the objects in the proposed regions. By using an object detector, VICE directly infers the position of each chess piece in an image. To map the detected pieces onto a chessboard, we model the corners of the chessboard as objects, for the object detector to detect simultaneously with the chess pieces, similar to the approach for dart scoring in (McNally et al., 2021). We use the detected corners of the board to estimate a homography matrix to project the image locations of each chess piece onto chessboard coordinates. This combines board localization and piece detection into one step, and avoids the need for geometry computations of the chessboard and granular location estimation of each square.

Due to chess's strict transition rules, we are able to infer the next state of the game given the previous state and current chess move. Since the initial state of a chess match is always fixed, we are able to track an entire match by tracking each move. Leveraging this information, VICE only detects the previous and new locations of chess pieces that moved. This omits the need for chess piece classification. Although this means later predictions depend on the current prediction, it is often easy to detect when an error is made — the current predicted move or a later predicted move will become illegal. Once the error is found and corrected manually, the subsequent moves will be updated automatically. Furthermore, as in Fig. 1, it is common in speed chess for a player to start a move before their opponent releases a chess piece. This can lead to game states where a hand fully occludes certain chess pieces in every frame. In these cases, without prior information, it is impossible to accurately infer the game state from a single image. In the case that the user does not want to track a match from the beginning, they can manually enter an initial state. Almost all chess digitizing methods include human interaction components (Wölflein and Arandjelović, 2021; Mallasén Quintana et al., 2021; Danner and Kafafy, 2015; Khan and Kesavan, 2014; Sokic and Ahic-Djokic, 2008; Gonçalves et al., 2005).

In summary, we introduce VICE as a prototype for a view-invariant chess estimator. By modeling the chess pieces that move as objects, VICE mitigates the effects of occlusion among chess pieces and from the human hand, situations that current methods struggle with. By modeling the corners of the chessboard as objects and detecting them simultaneously with the chess pieces, VICE is able to combine the chessboard

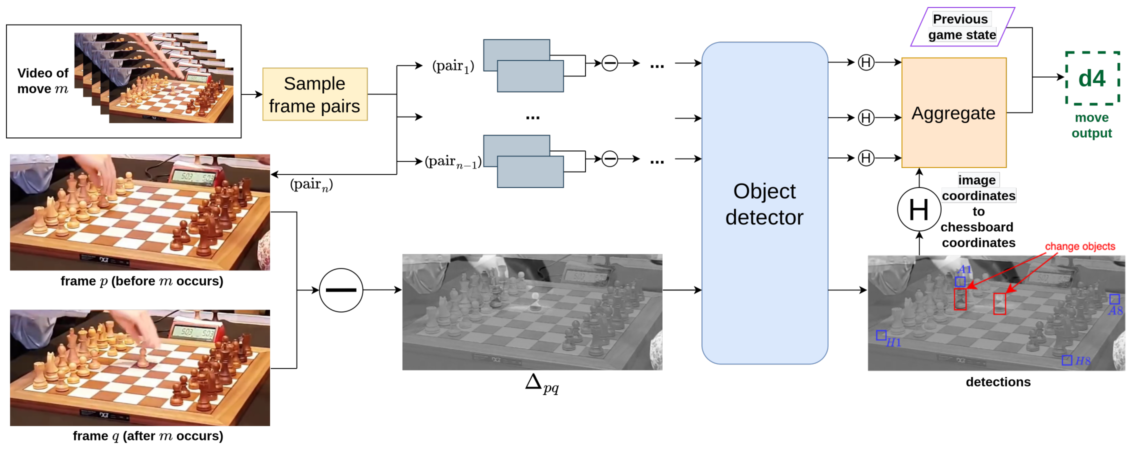


Figure 3: VICE samples frame pairs before and after a chess move and detects change objects (red) and corner objects (blue) from the image difference. Change detections are then projected onto chessboard coordinates by a homography matrix estimated by the corner detections. Detections from each sample pair are aggregated and combined with the previous game state to predict the current chess move.

localization and chess piece detection phases, making it the first single-step chess digitizing method. We focus on a simple and efficient design suitable for mobile deployment. We propose methods using simple geometry to (1) infer the bird's eye view center of a chess piece from its 2D bounding box and (2) overcome some performance limitations of small deep learning networks. We show that, compared to current chess digitizers, our prototype trained on minimal data generalizes better to unseen camera angles and is much smaller in model size.

2 RELATED WORK

2.1 Chess Digitization

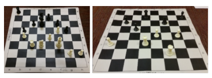
Vision-based chess digitization is often split into two phases – board localization and piece detection. Board localization is the process of estimating the image coordinates of each square on a chessboard. Piece detection aims to estimate the class of each chess piece and its location on the chessboard. Board localization is often done using traditional computer vision methods such as corner and edge detectors (Lu et al., 2015; Harris and Stephens, 1988). Early works on piece classification use hand-crafted features such as HOG (Dalal and Triggs, 2005) and SIFT (Lowe, 2004), then shifted to using CNNs with the development of deep learning. Currently, board localization methods (Mallasén Quintana et al., 2021; Czyzewski et al., 2021) are quite accurate when the conditions are good while chess piece classification remains the bottleneck for full chess digitization, mainly due to occlusion. A summary of recent methods are shown in Tab. 1.

Our work is similar to the methodology in (Hack and Ramakrishnan, 2014; Wang and Green, 2013; Matuszek et al., 2011; Sokic and Ahic-Djokic, 2008), where chess moves are identified by detecting the differentials between one movement and the next. This way, we omit the need for piece classification. However, in (Wang and Green, 2013; Matuszek et al., 2011; Sokic and Ahic-Djokic, 2008), a camera is placed directly over the board, which is not always feasible. CVChess (Hack and Ramakrishnan, 2014) addressed this perspective problem and aimed to digitize at a lower camera angle. The authors localized the board using Harris corner detectors (Harris and Stephens, 1988) and SIFT and predicted chess moves by computing the change in color of each square. However, CVChess requires an initial empty chessboard setup for board localization and does not take into consideration the case when a human hand occludes the board. Hence, CVChess is not applicable for the videos used in this study. Furthermore, CVChess's performance suffered from various factors such as shadows, changes in lighting, and occlusion among chess pieces.

2.2 Object Detection

Object detection is a fundamental task in computer vision that involves localizing and classifying objects in an image. Classical CNN-based object detectors are often divided into two-stage and one-stage detectors. Two-stage detectors, such as the R-CNN family (He et al., 2020; Dai et al., 2016; Ren et al., 2015; Girshick, 2015; Girshick et al., 2014), first generate

Table 1: Computer vision-based chess digitization methods.

Method	Description
ChessVision (Ding, 2016) 	**Board localization:** None. The user must manually select the four corners of the board, which are used to estimate a projection matrix. **Piece detection:** Support vector machines (SVMs) to classify chess pieces from SIFT (Lowe, 2004) features.
Neural Chess (Czyzewski et al., 2021) 	**Board localization:** 1) Canny detector (Lu et al., 2015) and CLAHE algorithm (Reza, 2004) to detect line segments, a handcrafted linking function to combine line segments, and an M-estimator-based algorithm (Wiens, 1996) to merge collinear lines. 2) Preprocess (apply grayscale, Otsu method (Jassim and Altaani, 2013), Canny detector, binarization) points where detected lines intersect and feed to a rule-based detector for simple cases or a CNN-based detector for other cases to determine lattice points. 3) An iterative algorithm to generate a heatmap from the detected lattice points that represents the likelihood of each pixel being the chessboard. **Piece detection:** Modified ChessVision by including physical properties of chess pieces in input, clustering similar pieces, selecting most probable piece configurations computed from a Stockfish chess engine (Stockfish, 2008). (A CNN implementation was also tested but showed minimal improvements over the previous SVM model.)
LiveChess2FEN (Mallasén Quintana et al., 2021) 	**Board localization:** Optimized Neural Chess to reduce cost and latency. **Piece detection:** CNN to classify the chess piece in each square, followed by an algorithm that uses chess rules as a constraint to refine the CNN output probability vector (e.g. cannot have 3 Kings on the board).
Chesscog (Wölflein and Arandjelović, 2021) 	**Board localization:** 1) Canny detector to detect lines. 2) DBSCAN clustering (Ester et al., 1996) to merge similar lines. 3) A RANSAC-based algorithm to determine lattice points from detected lines. **Piece detection:** Occupancy classification using a CNN to determine if each square contains a chess piece. Another CNN to classify the squares that contain a chess piece.

region proposals in the image, then classify the proposals in the second step. Single-stage detectors localize and classify objects in a single step, by making predictions with respect to pre-defined anchors (Redmon and Farhadi, 2017; Liu et al., 2016; Lin et al., 2020) or a grid of possible object centers (Tian et al., 2019; Zhou et al., 2019; Law and Deng, 2020; Redmon et al., 2016).

Transformer-based object detectors such as DETR (Carion et al., 2020) and its variants (Zhu et al., 2021; Liu et al., 2022; Li et al., 2022; Zhang et al., 2022) were recently proposed as end-to-end architectures that omit the use of hand-designed components such as spatial anchors and non-maximum suppression. Although improved classical detectors, such as Dyhead (Dai et al., 2021) and HTC (Chen et al., 2019) are still often regarded as the current best performing object detectors, it could be interesting to test transformer-based detectors in future work to see how well the attention mechanism and set predictions can capture the global structure of chess detections. In this paper, we chose to use tiny single-stage detectors for their computational efficiency and mobile friendliness.

3 SINGLE-STEP CHESS DIGITIZATION

To combine the board localization and piece detection phases, we model both the corners of the chessboard and chess pieces that moved as objects. We call the chess pieces that moved *change* objects and for notation, italicize "*corner*" to refer to the chessboard *corner* objects. We take the image difference of two video frames that occur before and after a move, and from which, detect the *change* and *corner* simultaneously using a lightweight object detector. The detected *corners* can then be used to compute a homography matrix that projects the detected *change* objects onto chessboard coordinates, which represent the previous and new locations of the chess pieces moved. This information, combined with the previous match state, allow us to infer the chess move and next match state (Fig. 3).

For each *corner* $K \in \mathcal{K} = \{A1, A8, H1, H8\}$, let $K^i \in \mathcal{K}^i$ be its image coordinates and $K^B \in \mathcal{K}^B$ be its chessboard coordinates. We set $A1^B = (0,0)$, $A8^B = (0,8)$, $H1^B = (8,0)$, $H8^B = (8,8)$. Let $\mathbf{H}$ be the homography matrix computed by mapping $\mathcal{K}^i$ to $\mathcal{K}^B$.

***Change* Objects:** are shown in the red bounding boxes in Fig. 4. They represent the previous and new locations of chess pieces that moved. In a chess match with M total moves, for any move $m \in \{1, ..., M\}$, let Π_m be the chess piece(s)[1] that moved. Let s_m be the frame a player's hand makes contact with the first chess piece that is being moved and e_m be the frame the player releases the last piece moved. Let p be a frame before move m, $(e_{m-1} < p < s_m)$ and q be a frame after move m, $(e_m < q < s_{m+1})$ where p and q have the same camera position. The red boxes represent the locations of Π_m in frames p and q.

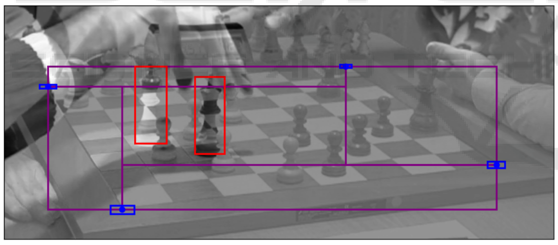


Figure 4: Bounding boxes of change objects (red), corners (blue), sides (purple).

For any *change* object c_j, let $c_j^i = (x_j^i, y_j^i)$ be its image coordinates and $c_j^B = (x_j^B, y_j^B)$ be its chessboard coordinates. We denote $c_j^B = \mathbf{H}(c_j^i)$, or more rigorously:

$$\begin{bmatrix} x_j^B \\ y_j^B \\ 1 \end{bmatrix} = \mathbf{H} \begin{bmatrix} x_j^i \\ y_j^i \\ 1 \end{bmatrix} \quad (1)$$

To ensure our method is invariant to different camera views and off-centered placements of chess pieces, we set the BEV center of c_j as its image coordinates. Since no 3D bounding boxes were labeled, we must infer the BEV centers from the 2D boxes. The prevalent method in 3D object detection is to tightly fit a 3D box inside the 2D box (Qin et al., 2022; Liu et al., 2019; Naiden et al., 2019; Mousavian et al., 2017). However, since the base of the chess piece is usually a circle, the bottom side of the tightly fitted 3D box will almost always intersect with the 2D box of the chess piece (see dashed blue square and red box in Fig. 5). Furthermore, due to the many plausible 3D boxes, the optimization steps are not only computationally expensive but also require the estimation of local yaw. Lastly, we only need the bottom center of the 3D box and not all 8 vertices.

Instead, we project the 2D box using the same homography matrix $\mathbf{H}$ and fit a circle tangent to the 3 sides (that does not include the top side) of the projected 2D box (Fig. 5). We take the center of this circle as the BEV center in chessboard coordinates c_j^B. Then the BEV center in image coordinates is simply $c_j^i = \mathbf{H}^{-1}(c_j^B)$. The center of this circle can be found by intersecting the two angle bisectors of the 3 sides of the projected 2D box. We make the assumption that the bottom side of the 3D box that bounds each chess piece is approximately square, which is usually the case for most chess sets.

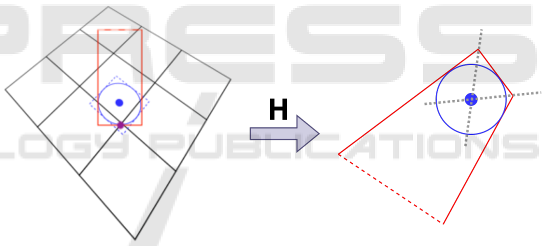


Figure 5: (Left): 2D bounding box of a chess piece in red. Top side of the 2D box in dashed red. Base of the chess piece in blue circle. The rest of the chess piece is not drawn. Although the center of the bottom side of the 2D box (purple dot) is usually tangent to the chess piece, using it to infer the chess piece's board location could result in the wrong grid, based on the placement of the chess piece and camera angle. Instead it is better to use the BEV center (blue dot). (Right): 2D box and chess piece base projected by homography matrix H. Since the 3 sides (not including the top side) of the 2D box tightly bound the circle, the BEV center can be found by connecting the 2 angle bisectors (grey dotted lines) of the 3 sides.

***Corner* Objects:** are shown in the blue bounding boxes in Fig. 4, where the center of each box corresponds to the image coordinates of a *corner* K^i. To account for depth, during labeling, we set the width and height of each box based on how far each corner can shift in a direction before producing a homography matrix that leads to incorrect chess moves.

For *corner* K, let $\mathbf{H}_r^K$ be the the homography ma-

[1] Multiple chess pieces could be moved, such as during castling.

Table 2: Accuracy (Acc.) refers to the percentage of moves correctly inferred by VICE on unseen matches, under 4-fold cross-validation, where each match is left out as the test set. 2-stage Acc. refer to the case where the corners are assumed to be known. Mean corner error (MCE) is calculated in terms of Euclidean distance in pixels, grouped by correct and incorrect move predictions (T; F). †- results shown for Match 2 are inferred after the detected corner labels were corrected by rotating 180° in the BEV plane (A1 should be H8). ‡- Match 4 starts at move 2 since the camera position is inconsistent during the first move.

(Detector)	Match 1: World Rapid 21 Carlsen, M. *vs.* Firouzja, A.	Match 2: World Blitz 19 Jobava, B. *vs.* Carlsen, M.	Match 3: Tata Steel India 18 Praggnanandhaa, R. *vs.* Mamedyarov, S.	Match 4: Tata Steel India 19 Ding, L. *vs.* Aronian, L.
(YOLOX-nano) Acc.	**81.0%**	**74.8%**[†]	72.3%	71.8%
MCE (T; F)	8.6; 9.9	9.7[†]; 11.0[†]	10.3; 18.1	6.1; 18.9
2-step Acc.	88.6%	83.8%	85.1%	84.5%
(NanoDet-Plus-m) Acc.	66.7%	33.3%[†]	**74.3%**	**73.2%**
MCE (T; F)	17.6; 31.3	15.3[†]; 218.8[†]	10.7; 27.8	4.2; 15.9
2-step Acc.	94.3%	84.7%	91.9%	87.3%
(YOLOv5-nano) Acc.	71.4%	-	45.3%	47.9%
MCE (T; F)	8.2; 17.3	-	6.8; 76.4	9.0; 37.9
2-step Acc.	84.8%	67.6%	83.1%	77.5%
Total moves (435):	*105*	*111*	*148*	*71*[‡]

trix estimated after shifting K^i to the right by r pixels with the other three corners fixed. Since we only require the integer value of the chessboard coordinates to infer the chess move, we set $r_K = \max(r)$ such that $\text{ceiling}(\mathbf{H}_r^K(c_j^i)) = \text{ceiling}(\mathbf{H}(c_j^i))$ is true for all c_j. Similarly, denote l, u, d as the left, up, down directions respectively. We set $w_K = \min(l_K, r_K)$ as the width and $h_K = \min(u_K, d_K)$ as the height of the bounding box for K.

If a *corner* class is not found during inference, the homography matrix can not be estimated, and the chess move will not be inferred. To remedy this, we also detect the four *sides* of the chessboard. The bounding boxes for each *side* (purple in Fig. 4) are created by simply using the two neighboring chessboard *corners* as opposite diagonal corners of the bounding box so no additional labeling is required. The combination of *corner* and *side* detections allows us to infer and localize the chessboard with missing detections. We found *side* objects to be more robust as there were far fewer missing *sides* than missing *corners* during inference (see Tab. 4).

4 EXPERIMENTS

We annotated 4 YouTube videos (ChessBaseIndia, 2022; ChessBaseIndia, 2020b; ChessBaseIndia, 2019; ChessBaseIndia, 2020a) of different chess matches with different camera angles and lighting and evaluate VICE's ability to digitize unseen camera angles using 4-fold cross-validation. In each fold, we leave out a different chess match as the test set and train a lightweight single-stage object detectors on the remaining 3 matches to detect the *corner*, *side* and *change* objects. We compare the performance of two keypoint-based detectors, NanoDet-Plus-m (Lyu, 2021) and YOLOX-nano (Ge et al., 2021), and one anchor-based method, YOLOv5-nano (Jocher, 2020). All runs are trained for 50 epochs, using the default training settings and data augmentation configurations for each object detector. Results are shown in Tab. 2. Implementation is detailed below in Sec. 4.1, 4.2.

4.1 Frame Pair Sampling

For chess move m, we sample frames p and q that occur before and after m and have the same camera angle, where $e_{m-1} < p < s_m$ and $e_m < q < s_{m+1}$ to create frame pair (p, q). For the first and last moves, we let $e_0 = s_1 - 5$ and $s_{M+1} = e_M + 5$.

To ensure our method is able to deal with human hands occluding the chessboard, we sample 4 types of frame pairs, $\mathcal{F}_i$:

- $\mathcal{F}_1 = \bigcup_j (p_1^j, q_1^j)$ with all possible combinations of $p_1^j \in \{s_m - 1, s_m\}$, $q_1^j \in \{e_m, e_m + 1\}$.
- $\mathcal{F}_2 = \bigcup_{j,k} (p_2^j, q_2^{j,k})$ contains at most 6 pairs. For each $p_2^j \in \{s_m - 1, s_m\}$, randomly sample 3 times: $q_2^{j,k} \in \{e_m + 2, \ldots, s_{m+1}\}$, $k = 1,2,3$.
- $\mathcal{F}_3 = \bigcup_{j,k} (p_3^{j,k}, q_3^j)$ contains at most 6 pairs. For each $q_3^j \in \{e_m, e_{m+1}\}$, randomly sample 3 times: $p_3^{j,k} \in \{e_{m-1}, \ldots, s_m - 2\}$, $k = 1,2,3$.
- $\mathcal{F}_4 = \bigcup_j (p_4^j, q_4^j)$ contains at most 6 pairs. For each $j = 1, \ldots, 6$, randomly sample: $p_4^j \in \{e_{m-1}, \ldots, s_m - 2\}$, $q_1^j \in \{e_m + 2, \ldots, s_{m+1}\}$.

Since a human hand is always partially occluding the board around frames s_m and e_m, sets $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3$ will always contain a hand. Set $\mathcal{F}_4$ will contain other random frames.

Next, we take the weighted image difference of a frame pair (p,q) as Δ_{pq}. Let I^j be the image at frame j. For frame pair (p,q), we have:

$$\Delta_{pq} = w_p I^p - w_q I^q \tag{2}$$

Since most chess sets contain chess pieces of just two colors, we convert Δ_{pq} to grayscale. In Tab. 4, we show that grayscaling improves the results. Lastly, we apply min-max normalization; hence the values of weights w_p, w_q are not important as long as $w_p \neq w_q$. In our experiments we set $w_p = 1.5$ and $w_q = 1$. From each Δ_{pq}, we detect the *change*, *corner* and *side* objects.

4.2 Inference

We use the center of the bounding box of each detected *corner* as its image coordinates. If a *corner* class is not found, we take the average of the 2 closest corners of the bounding boxes of the neighboring *sides*. Let S_{KL} be a side that connects *corners* K and L and let Γ_{KL} be the set of its 4 bounding box corners. For example, if *corner* K is not detected, we find $\gamma_{KL} \in \Gamma_{KL}$ and $\gamma_{JK} \in \Gamma_{JK}$ by minimizing $||\gamma_{KL}, \gamma_{JK}||_2$, and take mean$(\gamma_{KL}, \gamma_{JK})$ as the detected image coordinate for K (see Fig. 6: inferred *H8* in case 1 and inferred *H8* in case 2).

If a *corner* K is not found and only one neighboring *side* S_{KL} is detected with bounding box corner γ_{KL}^* closest to *corner* L (or is used to compute L), we set γ_{KL} as the detected image coordinate for K, where γ_{KL} and γ_{KL}^* are opposite diagonal corners of the bounding box of S_{KL} (see Fig. 6: inferred *A8* in case 1, inferred *A8* in case 2 and inferred *H1* in case 3). We denote all other situations as failure cases.

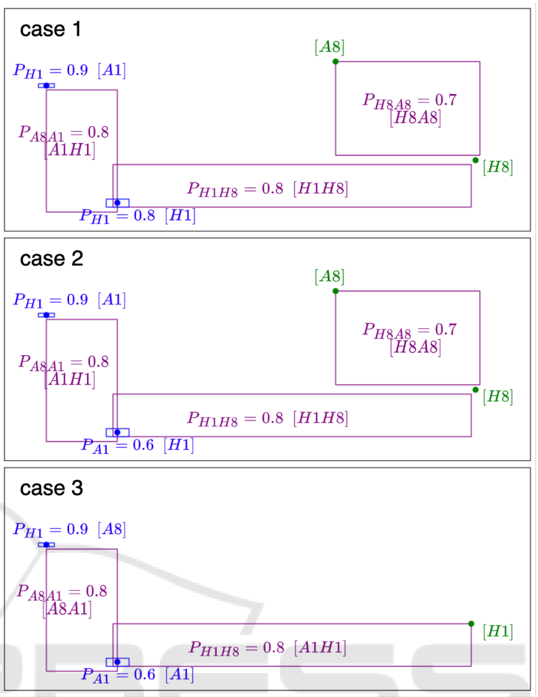

case 1
$P_{H1} = 0.9$ $[A1]$
$[A8]$
$P_{H8A8} = 0.7$ $[H8A8]$
$P_{A8A1} = 0.8$ $[A1H1]$
$[H8]$
$P_{H1H8} = 0.8$ $[H1H8]$
$P_{H1} = 0.8$ $[H1]$
case 2
$P_{H1} = 0.9$ $[A1]$
$[A8]$
$P_{H8A8} = 0.7$ $[H8A8]$
$P_{A8A1} = 0.8$ $[A1H1]$
$[H8]$
$P_{H1H8} = 0.8$ $[H1H8]$
$P_{A1} = 0.6$ $[H1]$
case 3
$P_{H1} = 0.9$ $[A8]$
$P_{A8A1} = 0.8$ $[A8A1]$
$[H1]$
$P_{H1H8} = 0.8$ $[A1H1]$
$P_{A1} = 0.6$ $[A1]$

Figure 6: Detected side bounding boxes in purple, where $P_\kappa = \rho$ indicates that the detection is of class κ with probability score ρ. Inferred class in square brackets. Similarly, detected corners in blue. Inferred corner locations in green. Case 3 is a failure case since the location of H8 cannot be inferred.

We found that, although the locations of the detections were adequate, the predicted class labels were inconsistent. This is likely due to the limited performance of tiny object detectors trained on minimal data. However, since the order of the *corner* labels of a chessboard is fixed in the counterclockwise direction (as long as the camera is above the chessboard), there are only 4 possible permutations of *corner* labels and we can "guess" which labels are misclassified based on the predicted locations.

We set the *center* of the chessboard as the mean of the bounding box centers of all predicted *corners* and *sides*. We take the detections with the highest probability scores and keep at most 4 *sides* and at most 4 *corners*, ensuring that the angle between neighboring *sides* with respect to the *center* is at least $(\frac{180}{8})^\circ$, and at most 1 *corner* can be between two *sides*. Then we check all 4 possible permutations of true labels and select the one that has the most matches with the predicted labels (Fig. 6 case 1 and 3). If there is a tie,

Table 3: The top performing versions of LiveChess2FEN and Chesscog shown. Total moves inferred are computed from rows 1, 4, 7 from Tab. 2.

Method	Version (backbone/head)	Size (MB)	Moves inferred
LiveChess2FEN (Mallasén Quintana et al., 2021)	(Xception)	176.4	-
Chesscog (Wölflein and Arandjelović, 2021)	(InceptionV3)	142.7	-
VICE (ours)	(YOLOX-n)	**1.8**	**75%**
	(NanoDet-P-m)	2.3	62%
	(YOLOv5-n)	3.8	40%

we select the combination with the highest probability score sum among the matched labels (Fig. 6 case 2).

For frame pair (p,q), let C^i_{pq} be the set of its detected *change* objects with probability scores over 0.5 and $\mathbf{H}_{pq}$ be the homography matrix computed by the detected *corners*. We project all *change* objects onto chessboard coordinates using $\mathbf{H}_{pq}$ and take the ceiling as the set of detections D_{pq} for frame pair (p,q):

$$D_{pq} = \{\text{ceiling}(\mathbf{H}_{pq}(c^i_{pq}));\; c^i_{pq} \in C^i_{pq}\} \qquad (3)$$

Let $\mathcal{P}_m$ be the set of all frame pairs associated with move m and $\mathcal{D}_m$ be the set of all detections for move m:

$$\mathcal{D}_m = \bigcup_{pq \in \mathcal{P}_m} D_{pq} \qquad (4)$$

If $\mathcal{D}_m$ contains 2 unique detections, they are set as the previous and new locations of the chess piece that moved during m. If $\mathcal{D}_m$ contains more than 2 unique detections, we take the top 3 most frequent detections and check if the coordinates fit any castling cases. If so, we output the corresponding castling case as move m. If not, we take the top 2 most frequent detections to infer move m. We denote all other situations as failure cases.

4.3 Results

Rows 1, 4, 7 of Tab. 2 show the percentage of moves VICE correctly infers in each unseen match using 3 different object detectors, with YOLOX-nano outperforming the others. The YOLOv5-nano version was unable to produce reasonable corner detections for Match 2. Although the results show that VICE was able to generalize to matches of new camera angles with limited training data, the *corner* labels predicted in Match 2 were all off by 180° in the BEV plane (*A1* should be *H8*). Hence, the results[†] shown in Tab. 2 for Match 2 are from the corrected *corner* detections. In practice, checking all 4 sets of results generated by the 4 possible permutations is still reasonable since the other 3 sets will very likely produce illegal chess moves. Current state-of-the-art vision-based chess digitizers such as Chesscog (Wölflein and Arandjelović, 2021) and LiveChess2FEN (Mallasén Quintana et al., 2021) also require the user to input the orientation of the board. Future versions of VICE can mitigate this problem by increasing training data diversity and expanding the data augmentation configuration to improve generalizability. As a prototype, VICE only includes the minimal inference scheme described in Sec. 4.2. Future versions can implement a grouping network to produce a more robust set of *corners*.

We calculate mean corner error (MCE) in terms of Euclidean distance in pixels between the detected and true corners. Rows 2, 5, 8 of Tab. 2 show the MCE of frame pairs that correctly and incorrectly inferred chess moves. Small errors in *corner* detections could lead to a chess piece projected to a square that neighbors its true square. Other than checking for castling, VICE does not include further postprocessing of predictions. Future versions can add a simple module to check and remove predictions that produced illegal chess moves. A chess engine could also be connected to produce chess move probabilities to regulate the predictions. This is done in existing methods such as (Mallasén Quintana et al., 2021; Czyzewski et al., 2021).

Rows 3, 6, 9 of Tab. 2 investigate how well our method works under the classical two-step chess estimation approach. We assume the *corner* locations are known and evaluate the *change* detections only. We find that NanoDet outperforms the other object detectors in this case. The overall improvement in performance of the two-step approach reinforces the fact that much of the error arises from the *corners*. Future versions of VICE should optimize training parameters to focus more on *corner* detection.

In Tab. 3, we compare VICE to LiveChess2FEN and Chesscog, which are both much larger in size. Since the two other methods infer game states from images, we run their algorithms on all of the frames sampled to form the frame pairs in our study. If the

Table 4: Total missing corners"/sides are computed as the sum of corner/side classes not detected from each frame pair. If all frame pairs associated with a move have at least one missing corner detection, we say the move has missing corners and no move prediction can be generated (row 3). [corners] refer to the inferred corners described in Sec. 4.2 using the side detections. Results in this table are generated from using the YOLOX-nano detector.

	M1	M2	M3	M4
Total frame pairs	2262	2246	3148	1463
Total missing *corners*	1244	481	1519	871
Moves with missing *corners*	19	1	1	2
Total missing *sides*	0	128	24	480
Moves with missing [*corners*]	0	0	0	0
Grayscale Acc.	81%	75%†	72%	72%
RGB Acc.	63%	59%†	35%	48%

algorithm correctly identifies the game state in both frames of a frame pair, a move is correctly inferred. We use the version of LiveChess2FEN that uses a Xception backbone and Chesscog with an InceptionV3 backbone, since they were the top performing models in their respective papers. LiveChess2FEN was able to localize the board for 12.7% of the frames in Match 1 and 97.1% of the frames in Match 3 but was not able to correctly infer any game states. Chesscog was able to localize the board in 24.7% of the frames for match 3 but was not able infer the game states. Since (Wölflein and Arandjelović, 2021) requires certain perspectives of the board as input (either *A1* or *H8* are at the bottom left corner), we tested using both the original image as input and the image rotated 90°.

Lastly, rows 2-5 of Tab. 4 show how the *side* objects helped with missing *corner* detections, as describe in Sec. 4.2. Future versions of VICE can incorporate global constraints such as (Gu et al., 2022) or set prediction approaches (Kuhn, 1955) to replace or complement the *side* objects. Rows 6-7 of Tab. 4 show the improved performance after grayscaling the image difference.

5 CONCLUSION

We introduced VICE, the first single-step chess digitizer that is capable of inferring chess moves from camera angles that current vision-based chess digitizers cannot. This allows chess players to automate the tracking of their in-person chess matches without the intrusiveness of a phone or camera placed directly above the chessboard, and is a cost-effective alternative to using electronic chessboards.

Our prototype is conceptually simple in design and focuses on computational efficiency and mobile friendliness. In Sec. 3 and Sec. 4.2, we presented methods using simple geometry to overcome some limitations in performance of tiny object detectors trained on minimal data. In Sec. 4.3, we showed that, compared to current vision-based chess digitizers, VICE generalizes better to unseen camera angles and is much smaller in model size. In Sec. 4.3, we discussed the shortcomings of our prototype and proposed how it can be improved with existing methodology.

ACKNOWLEDGEMENTS

We acknowledge financial support from the Canada Research Chairs Program.

REFERENCES

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-end object detection with transformers. In *Lecture Notes in Computer Science*, volume 12346 LNCS.

Chen, K., Pang, J., Wang, J., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Shi, J., Ouyang, W., Loy, C. C., and Lin, D. (2019). Hybrid task cascade for instance segmentation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

ChessBaseIndia (2019). Praggnanandhaa vs mamedyarov, tata steel chess india blitz 2018. *https://www.youtube.com/watch?v=ihzGKl2V4iE*.

ChessBaseIndia (2020a). Ding liren vs levon aronian, find the surprise move! tata steel chess india blitz 2019. *https://www.youtube.com/watch?v=sQcN9Zq3m-w*.

ChessBaseIndia (2020b). Magnus carlsen's killer move stuns jobavan, world blitz 2019. *https://www.youtube.com/watch?v=CUb76s1ZDb0*.

ChessBaseIndia (2022). Magnus carlsen vs alireza firouzja, full game, watch until the end, world rapid 2021. *https://www.youtube.com/watch?v=1WEyUZ1SpHY*.

Czyzewski, M. A., Laskowski, A., and Wasik, S. (2021). Chessboard and chess piece recognition with the support of neural networks. *Foundations of Computing and Decision Sciences*, 45.

Dai, J., Li, Y., He, K., and Sun, J. (2016). R-fcn: Object detection via region-based fully convolutional networks. In *Advances in Neural Information Processing Systems*.

Dai, X., Chen, Y., Xiao, B., Chen, D., Liu, M., Yuan, L., and Zhang, L. (2021). Dynamic head: Unifying object detection heads with attentions. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Proceedings - IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Danner, C. and Kafafy, M. (2015). Visual chess recognition. *https://web.stanford.edu/class/ee368/Project_Spring_1415/Reports/Danner_Kafafy.pdf*.

DGT (1998). Electronic chessboards. *http://digitalgametechnology.com*.

Ding, J. (2016). Chessvision : Chess board and piece recognition. *https://web.stanford.edu/class/cs231a/prev_projects_2016/CS_231A_Final_Report.pdf*.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*.

Ge, Z., Liu, S., Wang, F., Li, Z., and Sun, J. (2021). Yolox: Exceeding yolo series in 2021. *ArXiv*, abs/2107.08430.

Girshick, R. (2015). Fast r-cnn. In *International Conference on Computer Vision*.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Gonçalves, J., Lima, J., and Leitão, P. (2005). Chess robot system : a multi-disciplinary experience in automation. In *Spanish Portuguese Congress on Electrical Engineering; AEDIE: Marbella, Spain*.

Gu, J., Wu, B., Fan, L., Huang, J., Cao, S., Xiang, Z., and Hua, X.-S. (2022). Homography loss for monocular 3d object detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Hack, J. and Ramakrishnan, P. (2014). Cvchess: Computer vision chess analytics. *https://cvgl.stanford.edu/teaching/cs231a_winter1415/prev/projects/chess.pdf*.

Harris, C. G. and Stephens, M. J. (1988). A combined corner and edge detector. In *Alvey Vision Conference*.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2020). Mask r-cnn. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42.

Jassim, F. A. and Altaani, F. H. (2013). Hybridization of otsu method and median filter for color image segmentation. *ArXiv*, abs/1305.1052.

Jocher, G. (2020). ultralytics/yolov5: v6.2 - YOLOv5. *https://github.com/ultralytics/yolov5*.

Keener, G. (2022). Chess is booming. *The New York Times*.

Khan, A. M. and Kesavan, R. (2014). Design and development of autonomous chess playing robot. In *Int. J. Innov. Sci. Eng. Technol.*, pages 1–4.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 52.

Law, H. and Deng, J. (2020). Cornernet: Detecting objects as paired keypoints. *International Journal of Computer Vision*, 128.

Li, F., Zhang, H., Liu, S., Guo, J., Ni, L. M., and Zhang, L. (2022). Dn-detr: Accelerate detr training by introducing query denoising. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13619–13627.

Lin, T. Y., Goyal, P., Girshick, R., He, K., and Dollar, P. (2020). Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42.

Liu, L., Lu, J., Xu, C., Tian, Q., and Zhou, J. (2019). Deep fitting degree scoring network for monocular 3d object detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Liu, S., Li, F., Zhang, H., Yang, X., Qi, X., Su, H., Zhu, J., and Zhang, L. (2022). DAB-DETR: Dynamic anchor boxes are better queries for DETR. In *International Conference on Learning Representations*.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. In *Lecture Notes in Computer Science*, volume 9905 LNCS.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*.

Lu, X., Yao, J., Li, K., and Li, L. (2015). Cannylines: A parameter-free line segment detector. In *Proceedings - International Conference on Image Processing, ICIP*.

Lyu, R. (2021). Nanodet-plus: Super fast and high accuracy lightweight anchor-free object detection model. *https://github.com/RangiLyu/nanodet*.

Mallasén Quintana, D., Del Barrio García, A. A., and Prieto Matías, M. (2021). Livechess2fen: A framework for classifying chess pieces based on cnns. *ArXiv*, abs/2012.06858.

Matuszek, C., Mayton, B., Aimi, R., Deisenroth, M. P., Bo, L., Chu, R., Kung, M., Grand, L. L., Smith, J. R., and Fox, D. (2011). Gambit: An autonomous chess-playing robotic system. In *Proceedings - IEEE International Conference on Robotics and Automation*.

McNally, W., Walters, P., Vats, K., Wong, A., and McPhee, J. (2021). Deepdarts: Modeling keypoints as objects for automatic scorekeeping in darts using a single camera. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 4547–4556.

Mousavian, A., Anguelov, D., Košecká, J., and Flynn, J. (2017). 3d bounding box estimation using deep learning and geometry. In *Proceedings - IEEE Conference on Computer Vision and Pattern Recognition*.

Naiden, A., Paunescu, V., Kim, G., Jeon, B., and Leordeanu, M. (2019). Shift r-cnn: Deep monocular 3d object detection with closed-form geometric constraints. In *Proceedings - International Conference on Image Processing, ICIP*.

Qin, Z., Wang, J., and Lu, Y. (2022). Monogrnet: A general framework for monocular 3d object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Redmon, J. and Farhadi, A. (2017). Yolo9000: Better, faster, stronger. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition*.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*.

Reza, A. M. (2004). Realization of the contrast limited adaptive histogram equalization (clahe) for real-time image enhancement. *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, 38.

Sokic, E. and Ahic-Djokic, M. (2008). Simple computer vision system for chess playing robot manipulator as a project-based learning example. In *Proceedings of the 8th IEEE International Symposium on Signal Processing and Information Technology*.

SquareOff (2018). Intelligent chessboard. *https://squareoffnow.com*.

Stockfish (2008). Open source chess engine. *https://stockfishchess.org*.

Tian, Z., Shen, C., Chen, H., and He, T. (2019). Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE International Conference on Computer Vision*.

Wang, V. and Green, R. (2013). Chess move tracking using overhead rgb webcam. In *International Conference Image and Vision Computing New Zealand*.

Wiens, D. P. (1996). Asymptotics of generalized m - estimation of regression and scale with fixed carriers, in an approximately linear model. *Statistics and Probability Letters*, 30.

Wölflein, G. and Arandjelović, O. (2021). Determining chess game state from an image. *Journal of Imaging*, 7(6).

Zhang, H., Li, F., Liu, S., Zhang, L., Su, H., Zhu, J., Ni, L. M., and Shum, H.-Y. (2022). Dino: Detr with improved denoising anchor boxes for end-to-end object detection. *ArXiv*, abs/2203.03605.

Zhou, X., Wang, D., and Krähenbühl, P. (2019). Objects as points. *ArXiv*, abs/1904.07850.

Zhu, X., Su, W., Lu, L., Li, B., Wang, X., and Dai, J. (2021). Deformable detr. In *International Conference on Learning Representations*.