Light Cascaded Convolutional Neural Networks for Accurate Player Detection

Keyu Lu[1]
keyu.lu@nudt.edu.cn
Jianhui Chen[2]
jhchen14@cs.ubc.ca
James J. Little[2]
little@cs.ubc.ca
Hangen He[1]
hangenhe@nudt.edu.cn

[1] College of Mechatronic Engineering and Automation
National University of Defense Technology
Changsha, China

[2] Department of Computer Science
University of British Columbia
Vancouver, Canada

Abstract

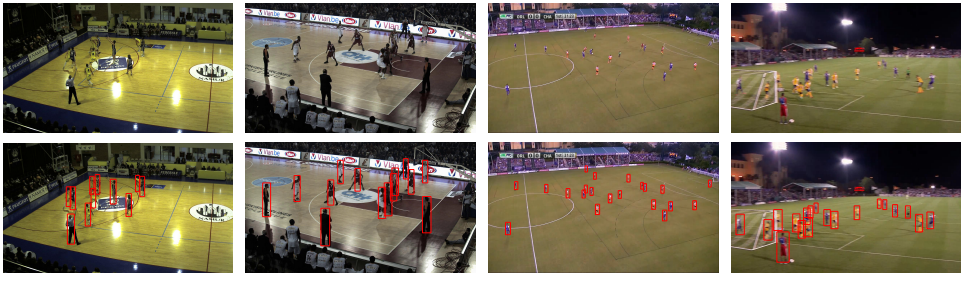
Vision based player detection is important in sports applications. Accuracy, efficiency, and low memory consumption are desirable for real-time tasks such as intelligent broadcasting and automatic event classification. In this paper, we present a cascaded convolutional neural network (CNN) that satisfies all three of these requirements. Our method first trains a binary (player/non-player) classification network from labeled image patches. Then, our method efficiently applies the network to a whole image in testing. We conducted experiments on basketball and soccer games. Experimental results demonstrate that our method can accurately detect players under challenging conditions such as varying illumination, highly dynamic camera movements and motion blur. Comparing with conventional CNNs, our approach achieves state-of-the-art accuracy on both games with 1000× fewer parameters (i.e., it is *light*).

1 Introduction

Player detection from images and videos is essential for a number of applications. For example, intelligent broadcast systems use player locations to guide the viewpoints of broadcasting cameras [1]. Furthermore, player detection provides metadata for player tracking, player pose estimation and team strategy analysis [2]. Player detection, as a subcategory of people detection, has been extensively studied. For example, background subtraction based methods [3, 4] have been applied to basketball games and achieved real-time response. However, these methods assume the camera is static or moves slowly so that they can robustly detect foreground objects. Some learning-based methods such as Faster R-CNN [5] and YOLO [6] can be also adapted to detect players with high detection accuracy but they may miss distant players because of low pixel resolution of these players.

Our work is inspired by CNN-based object detection and cascaded learning. Recently, a number of CNN-based approaches have achieved excellent performance for general object detection [5, 6, 7, 8, 9, 10, 11] or pedestrian detection [12, 13]. However, to the best of our knowledge, there are few deep neural networks specifically designed for player detection.

[1] This work was performed while Keyu Lu was visiting the LCI lab at UBC.



(a) Varying appearance (b) Cluttered background (c) Distant players (d) Motion blur

Figure 1: **Examples of player detection challenges.** The second row shows the detection results of our approach.

Compared with pedestrian detection, player detection is more challenging in terms of body and camera motions [14]. For example, basketball players quickly jump, extend their hands and twist their bodies (see Figure 1). Moreover, the heights of players vary from several pixels to hundreds of pixels so that conventional deep neural networks often miss smaller players because the activation of smaller player vanishes at the end of a deep network.

We propose a neural network to solve the problems raised above. We first design a cascaded convolutional neural network (CNN) for player/non-player classification. The cascaded CNN can quickly reject non-players in its shallow parts (i.e., early branches), greatly speeding up detection. We then present an end-to-end training approach to boost detection performance. Moreover, we apply a dilation strategy in testing to improve the detection accuracy. With these three techniques, our method can robustly detect players under dynamic camera views.

We are not the first to use cascaded CNNs [9]. However, our method is significantly different from previous methods. First, our method inherits feature maps from previous stages, which avoids recomputing the feature maps from the original image at every stage [9]. Second, our method jointly trains parameters in all stages via global optimization. It is significantly different from previous methods [9, 10] in which cascade stages are relatively independent. In addition, the main purpose of the dilated strategy in our method is to align feature maps instead of increasing the receptive field [15].

The main contributions of this work are three-fold:

- Design a cascaded CNN and a joint learning objective so that the learned model can quickly reject non-players in sports player detection. The trained model is very compact (less than 100KB) and is very efficient in testing (about 10 fps for images of 1280×720 with un-optimized Matlab code);
- Present a dilation strategy to achieve accurate player detection on various conditions such as varying appearance, dynamic camera movements and complex background;
- Propose a soccer player detection dataset which is available on-line[1] for research purposes. It contains numerous challenging situations such as varying illumination, player appearances, poses, zoom levels, motion blur, severe occlusions and cluttered backgrounds. This dataset is complementary to the APIDIS and SPIROUDOME datasets [16, 17] which focus on basketball games.

[1] http://www.cs.ubc.ca/~jhchen14/ccnn_player_detection/

We evaluate our approach on basketball and soccer games. In all experiments, our method can accurately detect players under challenging conditions.

2 Related work

Player detection Player detection from sport video has been addressed by a wide variety of methods in recent years. The most common approaches are based on background subtraction [3, 18, 19]. For instance, Zhong *et al*. [19] have introduced a domain-independent global color filtering method to extract player regions from background. In the same vein, Chang *et al*. [18] first estimate the dominant color of the court and then detect player candidates from the background. These approaches are efficient, but their performance can be easily affected by illumination changes, camera movements and the presence of spectators [14]. To make player detection more robust, researchers have developed stronger features such as histogram of oriented gradients (HOG) features [20] with the support vector machine (SVM) method [21, 22]. Moreover, researchers have combined different features such as edge [23], LBP [24] and motion [25] or have employed part-based model [26, 27] to improve performance. However, they are not as robust as deep learning based methods in general.

CNN-based object detection In recent years, deep learning has boosted the development of object detection. Convolutional neural networks (CNNs) [28, 29] stands out as one of the most competitive methods for object detection. For example, R-CNN [30] first employs Selective Search [31] to generate candidate bounding boxes (object proposals) and then applies CNNs to classify objects from these proposals. Thereafter, Fast R-CNN [7] and Faster R-CNN [5] have improved the performance of the region proposal based methods. Another set of approaches regard object detection as a regression problem. You only look once (YOLO) [6] and Single Shot MultiBox Detector (SSD) [8] are two well-known regression-based methods. Both of them can simultaneously output the bounding box, category and confidence score for each detected object. However, they require a large-scale CNN and yet their performance on small objects detection is unsatisfactory. To improve the performance of CNN-based methods, cascaded CNNs have been proposed to eliminate non-object samples step by step. Li *et al*. [32] have introduced a multi-resolution CNN cascade to quickly reject background regions for face detection. Angelova *et al*. [9] have used cascade deep nets and fast features for efficient pedestrian detection. More recently, Yang *et al*. [10] have constructed a cascaded architecture by applying discrete AdaBoost [33] after each convolutional layer.

3 End-to-end cascaded CNN

Figure 2 shows the pipeline of our method. It consists of a cascaded CNN for player/non-player classification, an end-to-end training approach for global optimization and a dilation strategy for accurate detection.

3.1 Neural network architecture

Our neural network has a cascaded architecture (See Figure 3). It contains a main network (dashed blue box) and four classification branches (dashed red box). The main network is modelled after AlexNet [28]. However, we use much fewer filters (16 or 32 *vs* up to 2,048) with a small size (3×3) in each layer. Each branch is a shallow and simple network that

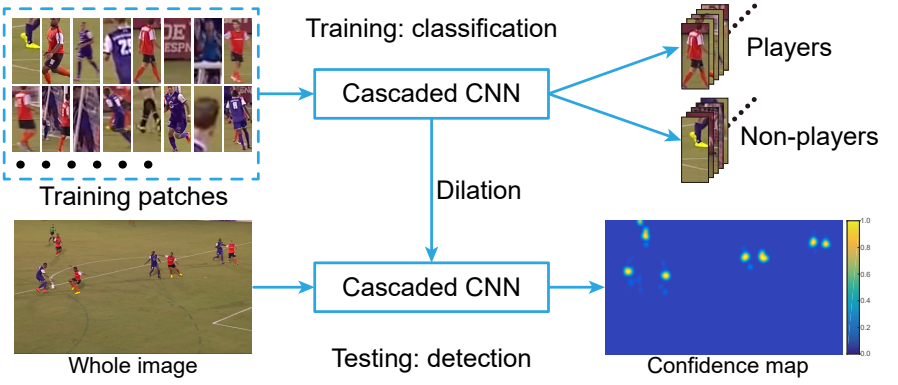


Figure 2: **Player detection pipeline**. We first train a neural network classifier from labeled image patches. Then, we detect player locations from a whole image using a dilation strategy. The network architecture, the training method and the dilation strategy are in Section 3.1, 3.2 and 3.3, respectively.

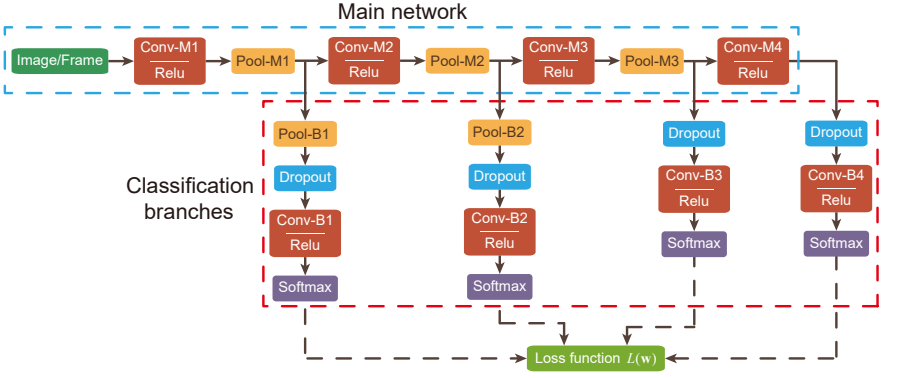


Figure 3: **Architecture of our cascaded CNN.** It has main network (blue box) and four branches (red box). The four branches are ordered (from left to right) to efficiently detect players. "-Mx" and "-Bx" stand for the x-th convolution or pooling layer in main network and classification branch, respectively.

classifies if the input has a player or not. As a result, the whole network is very light (less than 100 KB) compared with a conventional network which usually is more than 100 MB.

Our cascaded architecture fits the player detection task very well. For example, let us assume the input is an image patch. It will be passed to classification branches in order. If and only if the output of previous branch is positive (i.e., above an threshold), the image patch will go to next branch. By doing so, our network has two advantages. First, most negative examples are eliminated in the earlier branches so that our network is computationally efficient. Second, each branch can be trained for different levels of hardness of player detection. For example, branch one can eliminate most easy negative examples such as pure playing ground and leave hard examples to later branches. On the other hand, later branches (like branch four) can be specifically trained using hard examples. As a result, the whole network is more powerful when distinct branches work together.

Our network has an additional benefit that feature maps are partly shared in the main network. Because the main network and the classification branches are connected in a unified CNN framework, we can perform operations on the feature maps constructed by previous

stage instead of sampling image patches from the original image.

3.2 Network training

The training procedure has two steps: branch-level training and whole network training. The first step is crucial as it can significantly speed up the convergence rate of subsequent training process. In branch-level training, we regard each branch as an independent network. When the training of a particular branch is completed, we choose a recall rate threshold (97%) to remove negative samples that will not be used to train later branches. The four branches are trained in order using different sets of labeled samples.

When the branch-level training is completed, we perform end-to-end network training that simultaneously optimizes all the cascade branches. Our cascaded CNN is a cascaded classifier. Assume that the model has K cascade stages and is trained on N image samples. Let $D = \{(x_{i,j}, y_{i,j})\}$ denote the training set where $1 \leq i \leq N$ and $1 \leq j \leq K$. $x_{i,j} \in \mathbb{R}^d$ is the feature map of the i-th sample at the j-th cascade stage and $y_{i,j} \in \{0,1\}$ is the corresponding binary label. Then the probability for a sample to be positive is:

$$p_i(y_i = 1|\mathbf{x}_i, \mathbf{w}) = \prod_{j=1}^{K} p_{i,j}(y_{i,j} = 1|x_{i,j}, \mathbf{w}), \tag{1}$$

where $\mathbf{w}$ is the weights of the cascaded CNN. If a sample is classified as negative by any one of the cascade stages, our method predicts it as negative. Accordingly, we have:

$$p_i(y_i = 0|x_i, \mathbf{w}) = 1 - \prod_{j=1}^{K} p_{i,j}(y_{i,j} = 1|x_{i,j}, \mathbf{w}). \tag{2}$$

Inspired by [34], the loss function can be defined as:

$$L_p(\mathbf{w}) = -\sum_{i=1}^{N} [y_i \log(p_i(y_i = 1|x_i, \mathbf{w})) + (1 - y_i) \log(p_i(y_i = 0|x_i, \mathbf{w}))]. \tag{3}$$

In equations (1)∼(3), which stage first predicts an example as negative (negative elimination) does not affect the final result. However, fast negative elimination affects computational cost. For instance, if more negative samples are rejected by earlier cascade stages, fewer samples will be passed to upper stages, reducing the overall computational cost.

To achieve fast negative elimination, we design a regularization term for the loss function. Let T_j be the computational cost of the j^{th} cascade stage, which can be estimated according to the sizes of input feature maps, pooling and convolution kernels in this stage. Then, the regularization term is:

$$L_\Gamma(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{K} T_j \cdot \left[\prod_{u=1}^{j} p_{i,u}(y_{i,u} = 1|x_{i,u}, \mathbf{w}) > 0 \right], \tag{4}$$

where $[\cdot]$ is the Iverson bracket function which is activated only if the equation in it is satisfied.

The final loss function is a weighted function of accuracy cost and computation cost:

$$L(\mathrm{w}) = L_p(\mathbf{w}) + \beta L_\Gamma(\mathbf{w}), \tag{5}$$

where β is a weight to balance the accuracy term and regularization term. In this work, β is experimentally set to 0.5.

With this loss function, we train the whole cascaded CNN end-to-end using Adam algorithm [35]. Then, we estimate the cascade thresholds $\lambda = \{\lambda_1, \ldots, \lambda_K\}$ using a grid search over a range of thresholds. Only samples that satisfy $p_{i,j}(y_{i,j} = 1|x_{i,j}, \mathbf{w}) > \lambda_j$ can pass the j^{th} cascade stage.

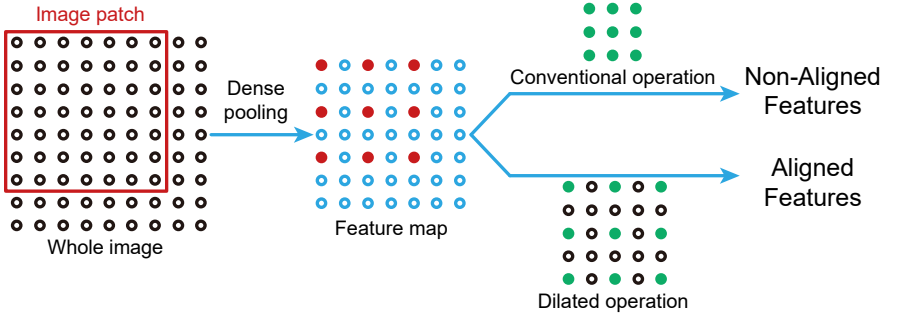


Figure 4: **The motivation of using the dilation strategy.** In the feature map (middle one) produced by dense pooling (trained with stride = 2), the feature points belonging to the red image patch are marked with red, which are separated by feature points generated by its neighbor image patches (marked with blue). In dilated operation (bottom right), valid positions (green points) are separated so that they properly align with the feature map generated by the previous layer.

3.3 Dilation strategy for accurate detection

Because the neural network introduced above is trained using image patches, directly applying it to a whole image would generate unaligned feature maps for operation kernels (see Figure 4). To address this issue, we develop a dilation strategy.

The dilation strategy aims to align feature maps before and after convolution and pooling layers in testing. Let us use the pooling layer as an example. In training, the pooling layer is 3×3 with stride $= 2$. In testing, the stride of the pooling layer should be changed to 1 to obtain an accurate feature location. As a result, the feature map will be misaligned before and after pooling (see Figure 4, top right), resulting in incorrect feature maps or requiring post-processing.

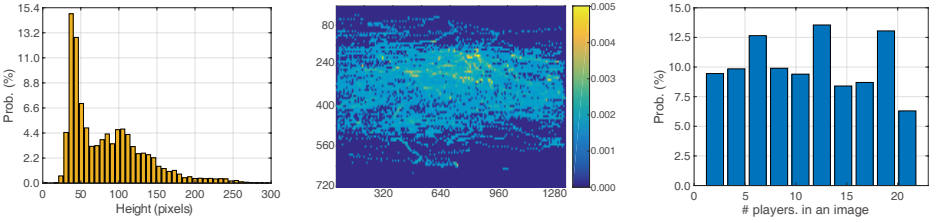
To address this problem, we employ a dilation strategy like [15] for convolution and pooling. For simplicity, we use 2D convolution as an example. Let w denote the convolution kernel (or weight) with size $m \times n$, each convolution step can be written as:

$$\eta = \sum_{i=1}^{m} \sum_{j=1}^{n} x_{i,j} \cdot w_{i,j}, \tag{6}$$

where x is the corresponding 2-D data of this convolution step. Let $\hat{w}$ and l be the dilated convolution kernel and dilation factor, respectively. The dilated version of convolution can be formulated as:

$$\hat{\eta} = \sum_{i=1}^{m} \sum_{j=1}^{n} x_{i,j} \cdot \hat{w}_{(i-1)\cdot l+1,(j-1)\cdot l+1}. \tag{7}$$

In the same way, dilated pooling can be performed by using the kernel with valid points located at $((i-1)\cdot l+1,(j-1)\cdot l+1)$. In dilated operation, as shown in Figure 4, valid positions are separated and the intervals of these positions are controlled by the dilation factor l. Consequently, the dilated kernel can spatially align with the separated feature map from dense pooling by adjusting the dilation factor. In this way, our neural network can be applied to the whole image without sacrificing detection accuracy. Moreover, the dilation strategy improves the efficiency of the method because it makes feature map sharing available in the whole image. More details are provided in the supplementary material.



(a) Players' heights (b) Players' centers (c) Number of players per frame

Figure 5: **Statistics of our soccer dataset.** The figures show the distributions of players' heights, centers and number of players per frame.

4 Experiments

4.1 Datasets and error metrics

Soccer dataset The soccer dataset is created from two professional matches hosted in the same stadium. Each match was recorded by three broadcast pan-tilt-zoom (PTZ) cameras (30 FPS and 1280×720 resolution). One camera is located at mid-field, overlooking the game. The other two cameras are located behind the left and right goal gates separately. Highlight video sequences were selected from the original video by professional editors. The highlight video represents typical soccer games events such as goals, goal attempts, passings and penalty kicks. 22,586 player locations were manually annotated from 2,019 images. The dataset contains numerous challenges such as varying player appearances, poses, zoom levels, motion blur, severe occlusions and cluttered background. Figure 5 shows the statistics of the dataset. The height of players, the image location of players and the number of players per image are widely distributed, demonstrating the diversity of the dataset. For example, the height of players is from about 20 pixels to 250 pixels with a long tail distribution from height of 150 pixels.

Basketball dataset We use two standard basketball datasets: APIDIS [16] and SPIROUDOME [17]. They were originally used for intelligent basketball video analysis [4, 36]. In these datasets, the challenges are from lower color contrast between players and background, and highly dynamic player movements.

In both datasets, 50% of images are randomly selected as training samples and the rest are used as testing data. For error metrics, we use an intersection-over-union (IoU) threshold of 0.7 to determine the correctness of detection.

4.2 Comparison with baselines

We build three baselines for the purpose of comparison. **Baseline 1** is a conventional CNN without a cascaded design. Table 1 top row (conventional) shows the structure of this baseline. **Baseline 2** is our method without end-to-end learning (i.e., with only branch-level training). **Baseline 3** is our method without the dilation strategy (Section 3.3).

Table 1 shows the comparison of our method with baseline 1. Our method is able to achieve a similar performance by using much less memory consumption (about $1000\times$). Although baseline 1 has no classification branches, it requires a large network to detect players from challenging scenarios. On the contrary, our method uses a cascaded network to reduce the complexity of the player detection problem and thus outperforms baseline 1 in terms of memory consumption. Table 2 shows the performance of our method in all stages. The accuracy is improved in every stage, especially from stage 1 to stage 2.

Type	Network Structure Conv-M1/B1	Conv-M2/B2	Conv-M3/B3	Conv-M4/B4	Memory	AUC
Conventional	64 / -	128 / -	256 / -	512 / 2	5.95 MB	0.873
	128 / -	256 / -	512 / -	1024 / 2	23.72 MB	0.937
	256 / -	512 / -	1024 / -	1024 / 2	58.61 MB	0.962
	256 / -	1024 / -	1024 / -	2048 / 2	81.10 MB	0.977
Cascaded (Ours)	8 /2	8 / 2	8 / 2	8 / 2	**12.46 KB**	0.881
	8 / 2	16 / 2	16 / 2	16 / 2	**31.84 KB**	0.932
	16 / 2	16 / 2	16 / 2	16 / 2	**38.18 KB**	0.967
	16 / 2	16 / 2	32 / 2	32 / 2	**79.43 KB**	0.973

Table 1: **Comparison between a conventional CNN and our cascaded CNN.** Our design achieves about 1000× memory (storage of network weights by single-precision floating-point) saving without sacrificing prediction performance. In this table, "Conv-Mx/Bx" stands for the number of filters in the x-th convolutional layer in the main network and its classification branch, respectively. All these filters have the size of 3×3. Detection performance is measured by area under curve (AUC) of receiver operating characteristic (ROC) curve on the soccer dataset.

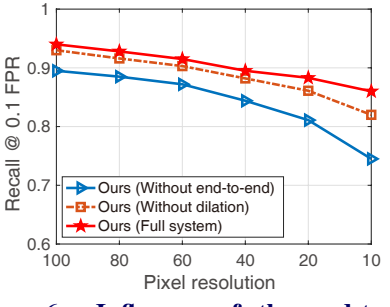


Figure 6: **Influence of the end-to-end learning and the dilation strategy.** It shows the recall @ 0.1 false positive rate (FPR) as a function of the players' pixel resolution on the soccer player dataset.

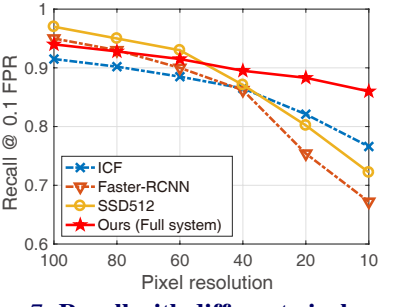


Figure 7: **Recall with different pixel resolutions.** The plot shows recall rate @ 0.1 false positive rate (FPR) as a function of players' pixel resolution on the soccer dataset.

Figure 6 shows the comparison of our method with the baseline 2 and the baseline 3 on the soccer dataset. Our method substantially improves the recall rate (at 0.1 false positive rate) in all levels of player resolutions.

4.3 Comparison with state-of-the-art methods

We compare the proposed approach with several state-of-the-art algorithms: BRF [4], ICF [37], Faster-RCNN [5] and SSD512 [8]. BRF [4] is a scene-specific classifier that is specifically designed for sport players detection. ICF [37] is a popular approach using hand-crafted features and the technique of integral channel features. Faster-RCNN [5] and SSD512 [8] both adopt CNNs for object detection, but the difference is that the former is a region proposal based method while the latter is a regression based method.

We conduct experiments on both the soccer dataset and the basketball dataset. To evaluate generalization capacity of these methods in player detection, we did cross game evaluation in which the model is trained on one game and tested on a different game without fine-tuning. There results are denoted by an extra "$-CRS$" (e.g. ICF$-CRS$).

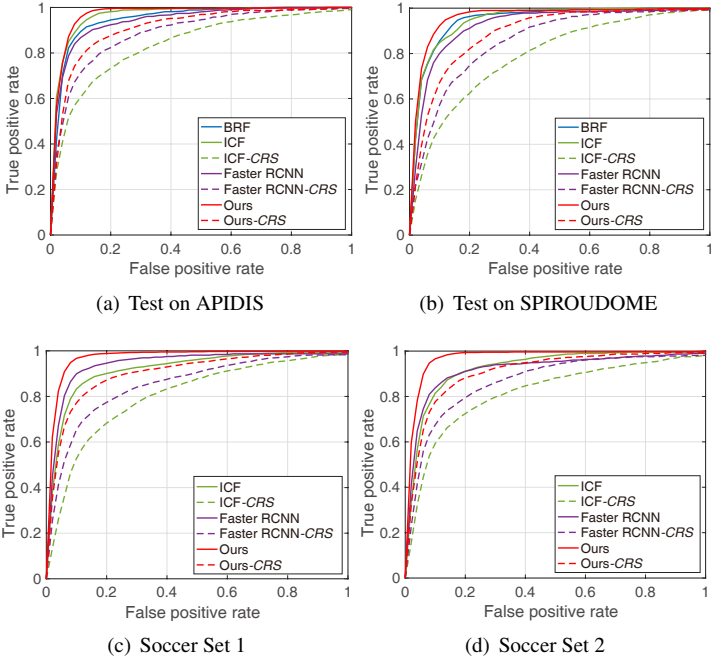
Figure 8 shows the receiver operating characteristic (ROC) curves of all the methods. For both datasets, our method achieves the best performance, with the performance gap being especially pronounced in soccer. In the cross games evaluation, our method (red-dash

Stage	Recall @ 0.1 FPR	Accuracy
1	0.991	0.463
2	0.973	0.872
3	0.967	0.931
4	0.962	0.944

Table 2: **Performance of all the stages on soccer dataset.** Non-player samples are rejected stage by stage.

Dataset	Cross game			Same game			
	ICF	Faster RCNN	Ours	BRF	ICF	Faster RCNN	Ours
APIDIS	0.838	0.887	**0.910**	0.950	0.968	0.941	**0.976**
SPIROUDOME	0.789	0.854	**0.889**	0.949	0.951	0.923	**0.965**
Soccer Set 1	0.802	0.850	**0.901**	-	0.918	0.938	**0.971**
Soccer Set 2	0.815	0.867	**0.908**	-	0.925	0.931	**0.969**

Table 3: **Performance comparison with state-of-the-art methods.** The detection performance is measured by areas under curve (AUC) of ROC curves. The best performance is highlighted.



(a) Test on APIDIS (b) Test on SPIROUDOME

(c) Soccer Set 1 (d) Soccer Set 2

Figure 8: **Performance evaluation.** These figures show the ROC curves of each method on player detection benchmarks. The curve closer to top left represents better performance.

lines) is also better than other methods, indicating higher generalization capability of our approach. The plotted lines of SSD highly overlap with the these of Faster-RCNN. Instead, we drew the performance of SSD in Figure 7 which has more space. Table 3 shows the areas under curve of the ROC curves. Our method is also substantially better than the second best Faster-RCNN method.

We also analyze the performance on players with different pixel resolutions in Figure 7. Our method is more robust than other methods when pixel resolutions decline. If we compare Figure 7 with Figure 6, we can find the performance gain in small players are from the end-to-end learning and the dilation strategy.

Figure 9 shows some qualitative results of our method. Our method performs robustly among various conditions. However, it may can not produce accurate bounding-boxes when players have complicated postures (e.g., the laying down players). One way to solve this problem is to integrate bounding-boxes regression strategy into our cascaded CNN, which

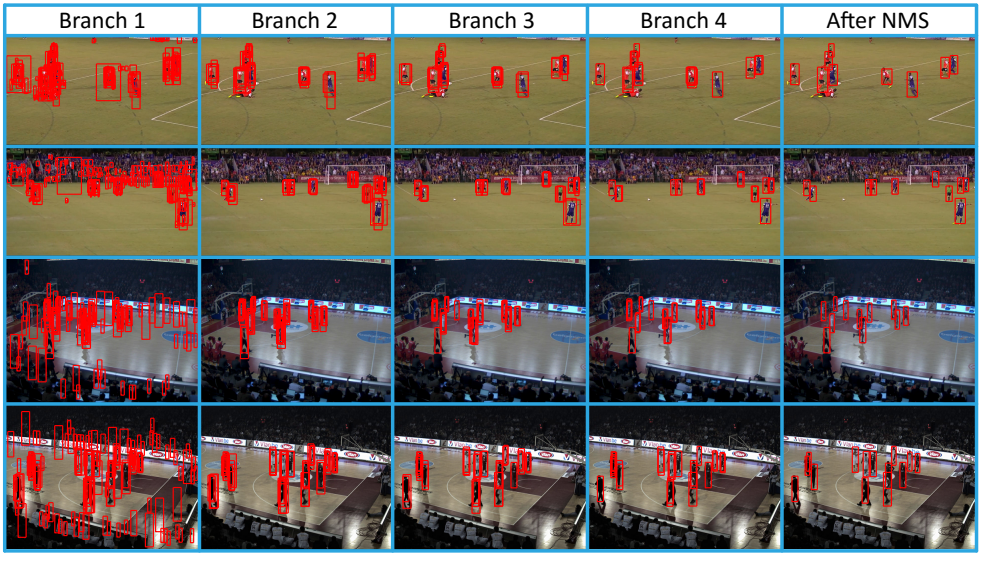


Figure 9: **Qualitative Results.** The proposed method is able to effectively reject non-player samples stage by stage and output accurate bounding boxes for players in both soccer and basketball games. In this figure, NMS stands for non-maximum suppression.

will be an interesting direction for future work.

4.4 Implementation

We perform experiments on a laptop with an Intel i7-6700HQ (2.6GHz) processor and a NVIDIA GTX1060 GPU. The detection speed is about 10 fps for images of size 1280×720 using un-optimized Matlab code based on MatConvNet [33]. Because our architecture achieves state-of-the-art performance while being much lighter (i.e., it has much fewer parameters) compared with conventional CNNs, it has the potential to be more efficient by optimizing the implementation.

5 Conclusions and future work

We have presented an accurate and efficient approach for player detection in group sports. We first introduced a light and effective cascaded CNN where all cascade stages are globally optimized end-to-end. Then we presented a dilation strategy to improve the detection accuracy of the network when performed on a whole image. In addition, we proposed a soccer player dataset to evaluate the robustness of player detection algorithms. Experimental results on both soccer and basketball datasets suggest that the proposed approach is light, effective and robust compared with many state-of-the-art detection methods.

Player detection has not been fully solved in terms of localization accuracy. In the future, we would like to employ regression strategies to obtain more accurate bounding boxes for players that have complicated postures. Our method is designed for applications with relative-simple backgrounds, which generally holds for sports applications. We have not test the method on varied backgrounds datasets such as Caltech pedestrian dataset, which we leave as a future work.

References

[1] J. Chen and J. J. Little. Where should cameras look at soccer games: Improving smoothness using the overlapped hidden Markov model. *CVIU*, pages 59–73, 2017.

[2] G. Thomas, R. Gade, T. B Moeslund, P. Carr, and A. Hilton. Computer vision for sports: Current applications and research topics. *CVIU*, pages 3–18, 2017.

[3] P. Carr, Y. Sheikh, and I. Matthews. Monocular object detection using 3D geometric primitives. In *ECCV*, 2012.

[4] P. Parisot and C. D. Vleeschouwer. Scene-specific classifier for effective and efficient team sport players detection from a single calibrated camera. *CVIU*, pages 74–88, 2017.

[5] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE TPAMI*, pages 1137–1149, 2017.

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016.

[7] R. Girshick. Fast R-CNN. In *ICCV*, 2015.

[8] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, and A. C. Berg. SSD: Single shot multibox detector. In *ECCV*, 2016.

[9] A. Angelova, A. Krizhevsky, V. Vanhoucke, and A. O. D. Ferguson. Real-time pedestrian detection with deep network cascades. In *BMVC*, 2015.

[10] F. Yang, W. Choi, and Y. Lin. Exploit all the layers: Fast and accurate CNN object detector with scale dependent pooling and cascaded rejection classifiers. In *CVPR*, 2016.

[11] H. Qin, J. Yan, X. Li, and X. Hu. Joint training of cascaded CNN for face detection. In *CVPR*, 2016.

[12] J. Liu, S. Zhang, S. Wang, and D. N. Metaxas. Multispectral deep neural networks for pedestrian detection. In *BMVC*, 2016.

[13] L. Zhang, L. Lin, X. Liang, and K. He. Is faster R-CNN doing well for pedestrian detection? In *ECCV*, 2016.

[14] M. Manafifard, H. Ebadi, and H. Abrishami Moghaddam. A survey on player tracking in soccer videos. *CVIU*, pages 19–46, 2017.

[15] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016.

[16] Apidis dataset. http://sites.uclouvain.be/ispgroup/index.php/Softwares/APIDIS. [Online; accessed Mar. 7th, 2017].

[17] Spiroudome dataset. http://sites.uclouvain.be/ispgroup/index.php/Softwares/SPIROUDOME. [Online; accessed Mar. 7th, 2017].

[18] M. H. Chang, M. C. Tien, and J. L. Wu. WOW: wild-open warning for broadcast basketball video based on player trajectory. In *ACM MM*, 2009.

[19] D. Zhong and S. F. Chang. Real-time view recognition and event detection for sports video. *Journal of Visual Communication and Image Representation*, 15(3):330–347, 2004.

[20] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[21] S. Mackowiak. Segmentation of football video broadcast. *International Journal of Electronics and Telecommunications*, 59(1):75–84, 2013.

[22] S. Baysal and P. Duygulu. Sentioscope: a soccer player tracking system using model field particles. *IEEE Trans. on Circuits and Systems for Video Technology*, 26(7):1350–1362, 2016.

[23] W. Xu, Q. Zhao, Y. Wang, and X. Li. Online learned player recognition model based soccer player tracking and labeling for long-shot scenes. *IEICE Trans. on Information and Systems*, 97(1):119–129, 2014.

[24] B. Li, C. Yang, Q. Zhang, and G. Xu. Condensation-based multi-person detection and tracking with HOG and LBP. In *IEEE International Conf. on Information and Automation*, 2014.

[25] M. Schlipsing, J. Salmen, M. Tschentscher, and C. Igel. Adaptive pattern recognition in real-time video-based soccer analysis. *Journal of Real-Time Image Processing*, pages 1–17, 2014.

[26] W. L. Lu, J. A. Ting, J. J. Little, and K. P. Murphy. Learning to track and identify players from broadcast sports videos. *IEEE TPAMI*, 35(7):1704–1716, 2013.

[27] Z. Ivankovic, M. Rackovic, and M. Ivkovic. Automatic player position detection in basketball games. *Multimedia tools and applications*, 72(3):2741–2767, 2014.

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[29] E. Walach and L. Wolf. Learning to count with CNN boosting. In *ECCV*, 2016.

[30] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[31] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. *IJCV*, 104(2):154–171, 2013.

[32] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua. A convolutional neural network cascade for face detection. In *CVPR*, 2015.

[33] Y. Freund, R. Schapire, and N. Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14:771–780, 1999.

[34] V. C. Raykar, B. Krishnapuram, and S. Yu. Designing efficient cascaded classifiers: tradeoff between accuracy and cost. In *ACM SIGKDD*, 2010.

[35] D. P. Kingma and J. L. Ba. A method for stochastic optimization. In *ICLR*, 2015.

[36] F. Chen and C. D. Vleeschouwer. Personalized production of basketball videos from multi-sensored data under limited display resolution. *CVIU*, 114(6):667–680, 2010.

[37] P. Dollár, Z. Tu, P. Perona, and S. Belongie. Integral channel features. In *BMVC*, 2009.

[38] A. Vedaldi and K. Lenc. MatConvNet – convolutional neural networks for matlab. In *ACM MM*, 2015.