Detecting Swimmers in Unconstrained Videos with Few Training Data

Nicolas Jacquelin[1,3], Romain Vuillemot[1,3], and Stefan Duffner[2,3]

[1] École Centrale de Lyon, Écully, France
[2] INSA Lyon, Villeurbanne, France
[3] LIRIS, UMR 5205 CNRS
{nicolas.jacquelin, romain.vuillemot, stefan.duffner}@liris.cnrs.fr

Abstract. In this work, we propose a method to detect swimmers in unconstrained swimming videos, using a Unet-based model trained on a small dataset. Our main motivation is to make the method accessible without spending much time or money in annotation or computation while maintaining performances. The swimming videos can be recorded from various locations with different settings (distance and angle to the pool, light conditions, reflections, camera resolution), which alleviates a lot of the usual video capture constraints. As a result, our model reaches top-performances in detection compared to other methods. Every algorithm described in the paper is accessible online at *https* : *//github.com/njacquelin/swimmers_detection*.

Keywords: Computer Vision · Swimming · Small Dataset · Segmentation · Detection

1 Introduction

Regarding swimmer analyses from competition videos, the main and most important aspect is to detect the swimmers in each frame. Although it is often not the terminal goal, a precise and reliable detection and localisation allows to perform further studies on swimming quality, helping the swimmers to improve their overall performance in a non-intrusive way. Therefore, a robust algorithm to effectively detect all swimmers of a race in a video is of utmost importance for both the athletes and their coach.

1.1 Problem Formulation

Swimmer detection in videos is the process of identifying the visible parts of the body in a picture. Detecting a swimmer in an image - and by extend in a video - may seem like a relatively easy task as state-of-the-art methods reach excellent results for human detection. However, the visible features in the environment of a pool are very different from those of daily-life walking and standing persons. A swimmer is mostly hidden under small and noisy waves, so a precise detection is much harder than for a normal human detection task. Therefore, an entirely

different model has to be created to detect swimmers during competitions and training.

This task is far from trivial, due to the complexity of the environment: the pool. It is full of reflections and diffraction, affected by unpredictable waves creating many local deformations in the image. Moreover, the light on the water tends to saturate the camera sensor, or at least obfuscate the swimmers underneath. Apart from that, even recent deep learning methods [13, 14, 11] usually require a large amount of carefully labelled data which are not easily available. However, we argue that a dataset with tens of thousands of images is not accessible to everyone to train their data on due to their computational cost and time (not everyone has access to GPU servers).

This paper shows a way to alleviate these problems without requiring thousands of labelled images, making the detection process work even with rough, non-expert annotations. Our approach could be applicable to detect unusual objects which are not in common detection datasets or daily-life context (with unusual visual features). The main contributions of this paper are the following:

- a model for automatic and robust swimmers detection in competition videos outperforming state-of-the-art large-scale object detection models,
- a annotated swimmers dataset,
- a method to easily train the model which reaches high performances with few data.

1.2 Related Work

Several past research works tackle the problem of swimmers detection with a computer vision approach. First, we need to mention the work of Benarab et al.[3, 2, 1], who recently proposed several techniques pursuing the same detection goal as ours. An interesting aspect of their work is the absence of deep-learning methods: their computer vision algorithms are mostly based on color gradient and low-level techniques. Although it allows a faster speed inference, this choice leads to hand-crafted, pool-specific thresholds and a lack of overall generalisation (they always use the same 2 races as an example for their papers). In the end, they do not provide a model or a metric to compare results.

Woinoski et al.[16] proposed a method based on a swimmer dataset annotated by themselves on a selection of 35 race videos, collecting about 25 000 images in the process. Sadly, it was not released before the end of our work, so we could not use it. However, creating such a data collection is a great milestone for the community, and the authors also applied current object detection methods on these data. As a result, they provide the mAP 25 (mAP definition in Section 3) of their best model, based on Yolo-V3 [13], which reaches 70% (Section 5, Fig.3 of [16]).

Hall et al.[9] propose a similar work, with an even larger dataset containing 327 000 images from 249 different races, but it appears they did not publish it. Their model is a 2-stage refinement method. They used tracking-based metrics, as opposed to the detection-based metrics we used.

1.3 Motivation

Detecting swimmers is an important part in automatic swimming video analysis, and it provides the basis for further, more detailed analyses.

Indeed, to estimate the swimmers positions in a pool from a race video, the detection in the 2D images is obviously required. Then, after a geometric projection into the coordinate system of the pool, it is possible to know the position of the swimmers in the pool plane, *i.e.* their distance in meters to the starting blocks and their lane. This whole detection and transformation process informs the swimmers relative position and rank, as well as their speed, speed variation, lap time etc. Furthermore, the number of swimming strokes during a length is also a metric of utmost importance to measure the swimming quality. To extract this information automatically, one needs to crop a sub-region surrounding them first, which is doable with a detection model.

Apart from these objective and quantifiable metrics, subjective and qualitative observations made by coaches are extremely precious for a swimmer. To perform them efficiently, coaches often need zoomed videos around their swimmers: such videos could either be extracted manually, by annotating the position on each frame, or be computed automatically as we propose in this paper. The extracted regions of interest and the automatic metrics mentioned before could greatly help the trainers to guide their swimmers. Automating this task would save a large amount of time allowing the coach to focus more on the technical aspect of swimming.

A good model is an important objective, but obtaining one with few data is rather challenging. Nowadays, there are many datasets with thousands or even millions of images of common objects [7, 10], but some problems arise. First, the swimmer class is present in none of these public datasets. Second, not everyone can afford to train a model on such data collections since this is computationally expensive and time-consuming. Third, fine-tuning a robust generic model is not interesting if the distribution is too different from the daily-life context it was trained on, which is the case in swimming. Therefore, the use of small specialized well-crafted datasets gets more and more attention in many applications, especially if they allow the creation of good detection models. Finally, a detection model trained from few annotated data can then be enhanced by training with more unlabelled data, using common semi-supervised learning techniques like self-training [4] or knowledge distillation [8]. This new, better model could itself be used to get a better precision in the aforementioned applications.

2 Proposed Approach

State-of-the-art detection methods, like Yolo [12] and Faster-RCNN [14], require at least tens of thousands of precisely labelled images to train a model. Such a dataset is extremely time-consuming to create, and it requires a large amount of time to train a model in an optimal way.

In this paper, we propose an easily trainable real-time method that can be trained with a small dataset of 400 images. The input labels are bounding boxes,

and the output can either be bounding boxes too, or a segmentation of the swimmers in an image. Despite these conditions, our solution gives excellent results and is usable in many different environments (inside/outside, pool/free water) and for a large variety of acquisition conditions (see Section 3). Further, it could easily be applied to other swimming-based sports like water polo.

2.1 Dataset Construction and Data Augmentation

The creation of a pertinent dataset is extremely important in order to get generalizable detection results with machine-learning based methods. To this end, we selected 12 international level competitions with openly available race videos with various points of view. This gave the dataset a great range of angle and size variation. For each competition, 8 races were selected, to represent the 4 main swimming styles performed by the 2 genders. One frame of these videos was saved every 3 seconds, resulting in 403 different images with a maximum of ten swimmers, where at least one of them is visible.

The swimmers bounding box annotation was made with the open-access tool LabelImg [6]. The main difficulty was to be consistent: the extremities (legs and arms) are often hidden under the water and not visible, but the annotation process needs to always highlight the same body parts. If for some images, the bounding boxes stop where the skin is not visible, and for other images they stop where the legs actually are (even if we cannot see them), that will cause divergence during the model training. Here, the decision was to frame the visible parts only so that the boxes are as small as possible around the athlete.

The data from 3 pools out of the 12 was used as test data and the 9 remaining as training data. The resulting dataset, that we called $Swimm^{400}$, was composed of 80 test images and 323 train images.

In addition to this carefully created dataset, we propose a specific data augmentation strategy in order to increase the trained models performance. It will also compensate the low quantity of images in $Swimm^{400}$.

Zoom-in / zoom-out: the main augmentation was the zoom in and out (Fig 1, right column). Image crops are performed during training, such that the subjects occupy more space. Inversely, a neutral colour could be put around the reduced image, so that the swimmers look smaller. This helped the model to generalize its representation of swimmers independently from their size. $Swimm^{400}$originally contains swimmers at different distances from the camera, but this data augmentation increased this benefit even more. Note that to allow batch training, the images were all resized to (256, 256) pixels after augmentation.

Side-switch: another important augmentation was the side-switch, seen in Figure 1, bottom row, third column. This transform is extremely useful to avoid central overfitting: most images present in $Swimm^{400}$tend to center the swimmer. The side-switch puts them on the side, preventing the model to only detect instances at the center.

Others: apart from these two transforms, other more common methods where used to train the model. The random left-right flip generalizes the swimmer direction to the model, by giving them the same chance to face each side.

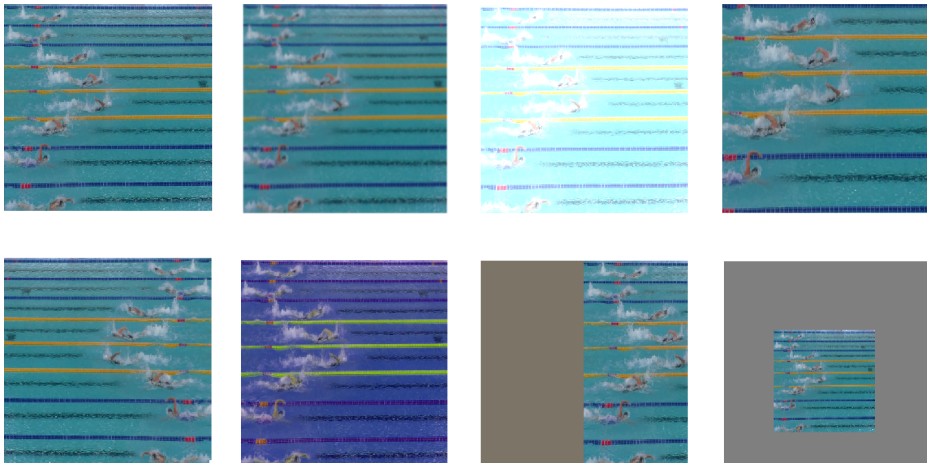

Fig. 1. The data augmentations used for the model training. From left to right, top to bottom: original image, blur, contrast and brightness change, crop, horizontal flip, hue change, side switch, zoom out.

The colour change (in HSV format, the hue is rotated by max. 45° so that the water can have any blue shade plus some green ones) generalizes to many skin, pool and water colors. The contrast and brightness random variations adapts the model to the many lighting conditions that can happen during a different competitions. Finally, Gaussian blur increases the overall robustness.

Of course, all these augmentations do not require any further annotation, as they are automatically generated during the training. The probability to trigger them is 50% each, except for color variation (30%) and side-switch (10%), as they both are stronger changes and thus might make the model diverge if used too much. These trigger probabilities work well for our study case, but may need to be slightly varied to adapt them to other detection problems.

2.2 Detection Model

We propose to train a relatively simple Convolutional Neural Network (CNN) using $Swimm^{400}$and data augmentation. It is designed to be deep enough to learn the complex feature hierarchies and patterns of high variability in our data, but not too much to overfit on the small dataset. Finally, the real-time constraint is important according to the experts we worked with, which discards Faster-RCNN [14] and similar methods with multiple sub-image inference.

Bounding box regression vs. segmentation: bounding box regression-based approaches [13, 14, 11] all rely on the same principle. They transform a part of the image into a semantic vector, from which the model computes the probability of presence and position of the objects on the image. Learning this transformation and providing stable results requires large amounts of labelled training images.

On the other hand, fully-convolutional models like Unet [15] simply transform each pixel into a "1" or "0" response according to the objective. This relatively simpler task brings two main advantages. First, it requires fewer data because the overall task is not regression (of the bounding boxes), but a classification extended to the whole image. Second, the conversion of a pixel into a presence probability amounts to a segmentation, which is actually a task at a higher level than just a bounding box regression. Indeed, according to the object position and orientation, much space contained inside a bounding box can be background, but most of a segmentation area designates the searched instance. Thus a segmentation model provides an alternative, more precise description of the regions of interest, as it has the possibility to exclude the parts of the surrounding background.

Tiny-Unet: we propose a variant of the well-known Unet architecture [15] for our swimmer detection model. The original model is a residual autoencoder with blocs of 3 convolutions layers with the same number of filters before a downsampling (in the encoder) or upsampling (in the decoder). The following modifications have been performed: instead of 3 convolution layers between each down/up-sampling, only one is performed. The filters are also smaller, increasing from 8 up to 128 instead of 64 to 1024 for the original Unet. We will refer to our model as *tiny-Unet*. Due to its shallow architecture and low filters number, it is able to run at 40 frames per seconds (FPS) on a GTX 1080 NVIDIA GPU, therefore the model is a real-time detector for 25 FPS videos.

To convert the model's output heatmap into bounding boxes, a threshold is applied to said heatmap, and the remaining areas are extracted. A bounding box is created by finding the circumscribed rectangle around each of them. This further allows for a fair comparison with the benchmark methods in Section 3, each of them creating bounding boxes.

Box-to-Segmentation-Map Transformation: the Unet model requires segmentation heatmaps for training. In order to convert the bounding boxes from $Swimm^{400}$into segmented data, an image with black background is created, and "filled" with white pixels inside the labelled boxes. Therefore, a pixel is 1 or 0 depending on whether there is or not a swimmer at the pixel. Multiple variants of this approach have been tested. Whiten inside the full box or only in the inscribed ellipsis; using smooth edges or hard edges. As shown in Table 1, the option giving the best result was to use the inscribed ellipsis with hard edges. As the boxes are reduced to approximate masks, we noticed that the model can be successfully trained even with mediocre and partly inconsistent annotation. This allows for a much quicker and less costly annotation process.

3 Experimental Results

First, we trained our tiny-Unet model on different variants of the box-to segmentation map strategy. The results are shown in Table 1.

This table clearly shows the superiority of shapes with hard edges. Smoothed ones tend to reduce the model convergence during training. Finally, filling an

Table 1. Detection performance of our model trained with different representations of the target heatmaps.

Training Data	mAP 25	mAP 50
Ellipse Hard Edges	**72**	**45**
Rectangle Hard Edges	60	28
Ellipse Smooth Edges	21	5
Rectangle Smooth Edges	13	3

ellipse shape is better than filling the whole rectangle bounding box. An intuitive explanation could be that corner regions are less likely to contain pixels from the instance. In fact, the ellipse mask contains almost only the swimmer, and the remaining pixels can be understood by the model as regularisation. As the edges of a swimmer are fuzzy anyway, it probably does not differ much from a precisely-labelled pixel-wise mask.

To compare the results of tiny-Unet with current state-of-the-art methods, we trained two variants of Yolo on $Swimm^{400}$. The first version is YoloV3 [13], a deep model with a 2048 fully-connected layer after the convolutions. The second is Yolo-tiny, which is shallower. Moreover, it replaces the fully connected layer with a 1×1 convolution layer with 56 filters, in order to drastically reduce the number of parameters to train.

The 3 models are trained with $Swimm^{400}$and the described data augmentation. The Adam optimizer is selected and starts with a learning rate of 10^{-3} with a decrease of 0.1 if the test loss plateaus more than 10 epochs. As the dataset is quite small, a batch size of 16 is chosen, and the loss is the Mean-Squared Error (MSE) in each case. For the Yolo-based models, the λ confidence training trick described in [12] Section 2.2 is followed.

To compare the models, the mean Average Precision (mAP) 25 and 50 are used, defined as:

$$mAP\ X = \frac{True\ Positives}{Positives}\ , \tag{1}$$

the "True Positives" being the detected bounding boxes with an Intersection over Union (IoU) of more than X% with the true box and "Positives" being the true number of boxes from the annotation. These metrics are not detection standards anymore (it is the COCO AP [5] which aims pixel perfect precision, which is nonsensical for swimmers with blurry edges under the water), but they are the closest to what real-world applications seek. Indeed, the precise delimitation of the swimmers' contour is not the priority here. The main objective of most applications is to obtain the overall position of the swimmers, for example to extract a sub-region around them. Thus, estimating the box barycenter and general size is enough, and therefore the mAP 25 and mAP 50 are good metrics. From Table 2, it appears that our tiny-Unet model outperforms by far the two others. Indeed, Yolo is a great model as long as enough data is available because of its conversion from feature vectors to output tensors. On the other hand, tiny-Unet does not require such a transformation. This result is confirmed by

Table 2. Performance comparison for the different detection models. They are all trained with the same data, except for the first line. In bold, the best of a category. We observe that the original UNET architecture gives significantly worst results compared to tiny-Unet, as it overfits on the few data.

Model	mAP 25	mAP 50
Yolo (from [16])	70	-
Yolo	24	12
Yolo-tiny	31	20
Unet	39	25
Tiny-Unet	**72**	**45**

testing the 3 models on the same videos : the tiny-Unet gives the best results. Moreover, with a video analysis, the tiny-Unet is much more stable between one frame and the next one.

The Yolo model trained on 25000 images from [16] gives results comparable to tiny-Unet trained on $Swimm^{400}$. It is not measured on the same benchmark though, thus we cannot assure which model exactly is superior. However, ours seems comparable to theirs with only a fraction of their amount of data and model size. In addition, it performs segmentation. Depending on the intended task, both the segmentation heatmap and the bounding boxes can be used, which is another advantage compared to the Yolo-based models.

Finally, tiny-Unet is extremely efficient in term of scalability. Being designed to be trained with small datasets, it is quite shallow and can run in real time (40 FPS) with not-so-recent GPUs (GTX 1080 NVIDIA GPU). Overall, the framework has been fairly optimized. The experts we interrogate can now the position of a swimmer in real time, but one swimmer at a time. With our detection model, we know the position of them all faster than real time, so more than 8 times faster than an expert, if there are 8 swimmers in the pool.

4 Discussions and Perspectives

The framework described in this paper is functional, but a few things can still be improved or modified to increase the overall performance. Also, despite having been proposed for swimming analyses, it can be generalized to other sports with low cost and small annotation time.

4.1 Improvements

The swimmers coordinates output by the framework could either be read as a segmentation area (the raw heatmap) or a bounding box (after the heatmap processing). The model is currently able to detect the parts of the swimmer that are outside of the water, but it does not detect any body part in particular. It results in an imprecise detection whose barycenter can vary from the shoulders to the hips. To alleviate this problem, we are currently creating another small

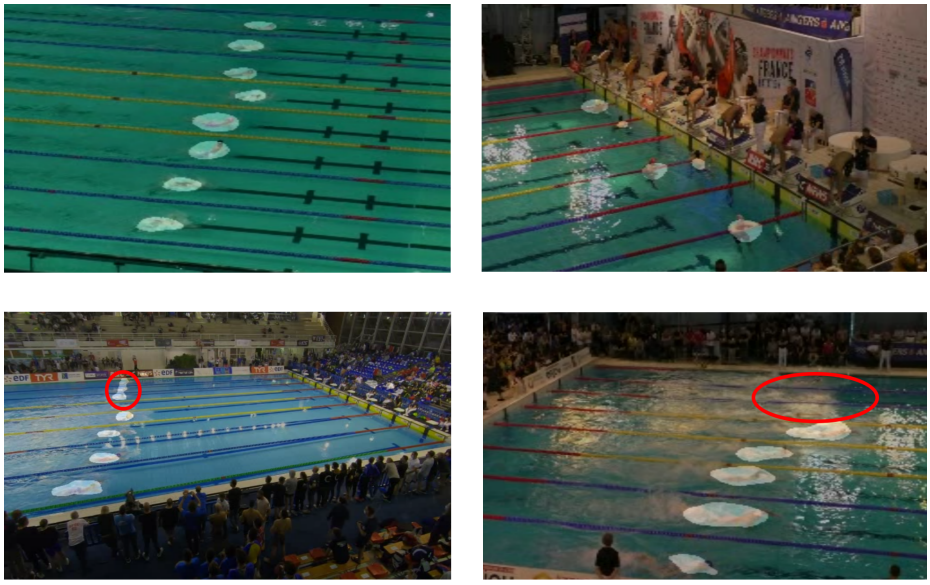

Fig. 2. The raw segmentation output overlaid on the input image. Top-left: the overall detection quality is very good. Top-right: even swimmers in the pool that are not swimming can be detected. Bottom-left: sometimes the blobs are too close and merge (circled in red); to solve this issue, the buoys lines are detected and they mask out the segmentation map, dividing the merged blobs into several. Bottom-right: the farthest swimmers are often not visible even by a human eye if the camera is too low compared to the pool (misdetections circled in red). Before filming, one should consider going as high as possible (within the limits of the room) to avoid this problematic behavior.

dataset with the same images as $Swimm^{400}$, but this time only the swimmers head is annotated. By training a model to detect the head only, we will obtain a higher robustness. This might be incorporated into a 2-stage detector similar to Faster-RCNN [14], the first stage being the raw detector described in this paper. Having this second stage will also potentially remove most of the few false positives. Figure 2 shows a few detection results of our model, and illustrates its limits.

4.2 Generalization to Other Sports

The detection process is quite general and easy to handle. For sports with atypical objects that are not present in usual datasets, our annotation and detection process can be reused and adapted. Indeed, the low quantity of data is enough for most detection tasks if the background does not change too much (a pool, a sport field etc.) as the model will more easily understand what actually matters. In our context we did not mention the rotation augmentation because it was not relevant. Though, it might be in other contexts.

Its performances could be improved with the creation of a bigger dataset (and then the use of a bigger model), but the main point of this paper was to prove that powerful specific analysis tasks can be achieved at low cost without large computational and human resources.

5 Acknowledgements

We thanks Renaud Jester form Ecole Centrale de Lyon and David Simbada & Robin Pla from the FFN (French Swimming Federation) for their contributions and expertise through project NePTUNE. This work was funded by the CNRS.

References

1. Benarab, D., Napoléon, T., Alfalou, A., Verney, A., Hellard, P.: A novel multi-tracking system for the evaluation of high-level swimmers performances. Baltimore, United States (May 2014)
2. Benarab, D., Napoléon, T., Alfalou, A., Verney, A., Hellard, P.: Optimized swimmer tracking system by a dynamic fusion of correlation and color histogram techniques. Optics Communications **356**, 256–268 (Dec 2015)
3. Benarab, D., Napoléon, T., Alfalou, A., Verney, A., Hellard, P.: Optimized swimmer tracking system based on a novel multi-related-targets approach. Optics and Lasers in Engineering **89** (05 2016)
4. Chapelle, O., Zien, A., Schölkopf, B.: Semi-supervised learning. MIT Press (2006)
5. COCO: Coco detection metric (2021), https://cocodataset.org/detection-eval
6. DarrenL: Labelimg (Dec 2018), https://github.com/tzutalin/labelImg
7. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255 (2009)
8. Gou, J., Yu, B., Maybank, S.J., Tao, D.: Knowledge distillation: A survey. International Journal of Computer Vision **129**(6), 1789–1819 (Mar 2021)
9. Hall, A., Victor, B., He, Z., Langer, M., Elipot, M., Nibali, A., Morgan, S.: The detection, tracking, and temporal action localisation of swimmers for automated analysis. Neural Computing and Applications **33**, 1–19 (06 2021)
10. Lin, T.Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C.L., Dollár, P.: Microsoft COCO: Common objects in context (2015)
11. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. Lecture Notes in Computer Science p. 21–37 (2016)
12. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection (2016), http://www.poker-edge.com/stats.php
13. Redmon, J., Farhadi, A.: Yolov3: An incremental improvement (2018)
14. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks (2016)
15. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation (2015)
16. Woinoski, T., Bajić, I.V.: Swimmer stroke rate estimation from overhead race video (2021)