# Detecting Swimming Pool using Aerial Imagery

Cruoglio Antonella  Iovino Giuliana  Mascolo Davide  Napoli Mario

**Abstract**

Object detection is one of the most important tasks in the field of Computer Vision. Locating a specific object in an image is a trivial task for humans, but can be quite challenging for machines. The goal of this project is to detect the swimming pool from aerial imagery using different Deep Learning techniques and we implement FasterRCNN, RetinaNet and Yolo. We experiment on a custom dataset publicly available on the Kaggle platform. We show that FasterRCNN model is better than RetinaNet and YOLO with our dataset focused on the detection of swimming pools.

## 1 Introduction

Tax assessors at local government agencies often have to rely on planimetric mapping services to create tax assessment rolls. Such surveys are expensive and infrequent, leading to inaccuracies in assessment of taxes. In the case of assessing property taxes, pools are typically added to assessment records because they impact the value of the property. This task would be the detection of swimming pools using aerial imagery which is a better solution. An application of this project was made in France in October 2021 from a collaboration between Capgemini and Google. In nine test regions, the tax department was able to detect more than 20,000 undeclared pools on aerial images thanks to the software, which should lead to tax revenues of about ten million euros.

## 2 Related Work

We use two different types of CNN based object detection algorithms: Single stage detector like YOLO and RetinaNet and Two stage detector like FasterRCNN.

### 2.1 FasterRCNN

The Region Proposal based framework is a two-step algorithm which gives a coarse scan of the whole scenario firstly and then focuses on regions of interests. This method can be divided into three different parts. The Region Proposal generation that is responsible for generating region proposals using selective search; the second part is a CNN used for features extraction in each region proposal and the third part is classification by SVM.

### 2.2 RetinaNet

RetinaNet is a single, unified network composed of a backbone network and two task-specific sub-networks. The backbone is responsible for computing a convolutional feature map over an entire input image. The first subnet performs convolutional object classification on the backbone's output; the second subnet performs convolutional bounding box regression.

### 2.3 YOLO

This method is an unified detector that uses features from the entire image to predict each bounding box. It requires only a single forward propagation through a neural network to detect objects and it also predicts all bounding boxes across all classes for an image simultaneously. The network divides the input image into an $S$ x $S$ grid, if the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts.

# 3 Data

We use the dataset available on Kaggle, that contains 3,197 annotated pools with different shapes and hues. The original tile size is 25,000 x 25,000 pixels. Then it is cropped into patches of 512x512 pixels without overlaps.

# 4 Experimental Results

The implementation of Faster-RCNN and RetinaNet are adapted using the pretrained version available with Pytorch library. We implement YOLO model using their respective papers and links of their github repositories. For Faster-RCNN we use ResNet50 as backbone and these parameters with an initial learning rate of 0.001, with weight decay of 5e-4 and momentum of 0.9. We use the SGD and the model is trained for 10 epochs on cuda (type Tesla T4), using a batch size of 8 and the images size of 416 x 416. We use different data augmentation techniques: flipping, random rotation, motion blur, median blur and blur. We apply popular metrics with the precision, recall and F1-Score as metrics. The final results are show in the Table 1.

| Model | Precision | Recall | F1Score | Time (train) |
|---|---|---|---|---|
| Faster-RCNN | 0.911 | 0.965 | 0.937 | $\sim 34\ min$ |
| RetinaNet | 1.00 | 0.745 | 0.853 | $\sim 33\ min$ |
| YOLO | 0.921 | 0.925 | 0.923 | $\sim 5\ min$ |

Table 1: Performance of models

We found that all the models have great performance except for the RetinaNet that has the lowest value of Recall and as we expected at the beginning of this project the FasterRCNN model is more time consuming with respect to the other two models. To better understand the implemented models, we visualize some object detected. We can see some examples of images used in FasterRCNN and good predictions Figure 1.


Figure 1: Good Predictions FasterRCNN.

In the Figure 2 we can see a comparison between FasterRCNN and YOLO. In particular the Faster RCNN generates more false positives than the YOLO model on this specific image.


Figure 2: FP in Faster RCNN and YOLO

In the figure 3 we can see a comparison between the three different models on the same image.The detection of this pool is not very simple because the size is small and it is partly covered by trees. As can be seen, the RetinaNet model is not able of detecting the pool unlike the Faster RCNN model which is more robust to occlusion, similarly to the Yolo model.


Figure 3: FN in RetinaNet

Of course, also if the Faster RCNN is the best model for this task it computes different errors and in the figure 4 we can see an example of False Positive.

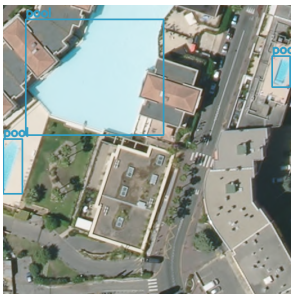

Figure 4: FP in Faster RCNN

# 5 Conclusions

We can conclude that all the models reach good performances, in particular we want to minimize false negatives maximizing Recall, at the same time we want to minimize false positives maximizing Precision and considering that we are interested in minimize both we can also look at F1 Score. We achieve the best Recall and F1-score with the Faster R-CNN model.

# 6 Future Works

We can implement DETR model that is a transformer end-to-end used for object detection but at the same time could be useful using a different set of parameters for the model already implemented in the work.

# 7 References

[1] Anthony T. Swimming pool detection from Aerial Imagery. Stanford University, Department of Computer Science.

[2] Jian D, Nan X, Gui-Song X, Xiang B, Wen Y, Michael Ying Y, Serge B, Jiebo L, Mihai D, Marcello P, Liangpei Z. Object Detection in Aerial Images: A Large-Scale Benchmark and Challenges.

[3] Jifeng D, Yi L, Kaiming H, Jian S. Faster R-CNN: R-FCN: Object Detection via Region-based Fully Convolutional Networks

[4] Shaoqing R, Kaiming H, Ross G, Jian S. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.

[5] Tsung-Yi L, Priya G, Ross G, Kaiming H, Piotr D. Focal Loss for Dense Object Detection, Facebook AI Research (FAIR).

[6] Joseph R, Santosh D, Ross G, Ali F. You Only Look Once: Unified, Real-Time Object Detection, University of Washington, Allen Institute for AI, Facebook AI Research.

[7] Zhong-Qiu Z, Peng Z, SHou-tao Z, Xindong W. Object Detection with Deep Learning: A Review.