Imperial College London

Schlumberger

MSc Applied Computational Science and Engineering
Department of Earth Science and Engineering

Inference of Technical Data from CCTV images

Author: Laura Su
GitHub alias: LCS18
email: lcs18@ic.ac.uk

Supervisors: Robert Zimmerman, Imperial College London
Adam Bowler, Schlumberger [1]
David Halliday, Schlumberger [1]

August 30, 2019

[1]Schlumberger Cambridge Research, High Cross, Madingley Road, Cambridge, CB3 0EL

Abstract

In Well Construction, CCTV data are used mainly for security and safety. In this thesis, the use of those same data to extract technical information, by monitoring tools or people, will be investigated. In particular, knowing that productive time includes "on-bottom time" (drilling) and "flat time" (necessary operations done when the drilling is not occurring), breaking down activities into those involving people and those not to identify that flat time and then identifying the associated sub-activities will help improving efficiency on the rig. By considering that during drilling the active part of the rig is void of people and that during the necessary operations human activity is involved, the density/number of people will be considered as an approximation for flat time. Determining if CCTV data can be used to analyse and optimise this flat time will be the main focus of the project.
To solve this problem, object detection of humans is implemented using Python with a Faster R-CNN + FPN architecture pre-trained on COCO dataset. The final output is the number of people depending on time, which is compared to the drilling state calculated from conventional drilling data.
It is shown that a correlation can be seen between those two approaches as it gives approximately the same ranges of time. Using CCTV can be considered as another inexpensive mean of breaking down periods of activities when classical instrumentation is deemed too costly.

Keywords: object detection, person, flat time, CCTV, drilling, well construction

Contents

1 Introduction

In many industrial settings, CCTV can be of value in monitoring employee safety and for site security. Advances in Computer Vision techniques can allow objects (people) to be detected and tracked automatically. Such techniques allow for the rapid analysis of large amounts of video data, allowing for quick intervention which may be critical for safety or security. This data could also provide valuable information on technical operations. This may include monitoring of equipment or monitoring of people when performing activities.

In the Well Construction setting, CCTV data, mainly used for safety and security, could also be used to improve efficiency on the rig. CCTV data is widely deployed and can constitute an inexpensive powerful tool as a complementary source of information for problems where classical instrumentation is deemed difficult or costly.

Breaking down the different stages of activity on the rig floor during the Well Construction process is another possible application of using CCTV data and will be the main subject this project will focus on. In Well Construction operations, productive time is typically split into "on-bottom time" when the drill bit is at the bottom of the hole, and the well is being progressed, and "flat time" when drilling is not occurring, but some other routine activity that is critical to the operation is being carried out.

CCTV data can be used to analyse flat time, as typically flat time will involve human activity on the rig floor, whereas on-bottom drilling would see the rig floor void of people. Thus, the density of people on the rig floor at a given time can be considered as a proxy for flat time.

Understanding the distribution of this flat time, and subsequently being able to optimise (minimise) this time is one way in which the efficiency of drilling operations can be improved.

This project will aim to use CCTV data from real drilling operations, to:

- Identify when and how many people are on the drilling rig floor
- Compute the distribution of people vs time, and compare with the drilling state calculated from conventional drilling data (weight-on-

bit, speed of the rotating drill-pipe (RPM), rate of penetration (ROP), torque, ...)

- Make a recommendation on the benefits (or otherwise) of the use of CCTV data to identify flat time.

This project is being conducted as part of a 6 month internship in the Data Analytics Group at Schlumberger Cambridge Research. This group has a focus on operational data, much of which is confidential. For the purpose of this report, public data from Youtube will be used to demonstrate the work.

As an extended goal, the project may also allow for the breakdown of the periods of activity into sub-activities, which may be used to identify areas where flat-time could be minimised.

2 Literature review

The aim is to get some knowledge about the current techniques used in the Oil and Gas industry for Well Construction and to investigate state-of-the-art methods in Computer Vision.

2.1 Normal drilling operations

The standard operations include:

- drilling the well
- adding a new joint of pipe as the hole deepens
- tripping the drill string (column of pipes) out of the hole to put on a new bit (tool that drills) and then running it back to the bottom of the well
- running and cementing the steel casing used to seal selected intervals of the hole

The book [5] gives a detailed description of all these operations which are the core of the project. According to this book, many steps are taken and repeated until completion of the well.

2.2 Improving well drilling

Improving well drilling processes and taking care of specific cases is still a research subject.

In pursuit of increasing quality and profitability, automating processes is a growing research subject. According to [12], the oil and gas industry tries to automate drilling to mimic the success of many other industries in automating processes. It would help performing faster smoother drilling and it would help controlling the process accurately. However, a full automation of drilling is difficult to put into practice as some operations still rely on human judgment and experience. Thus, automation is currently still a research topic.

Also, advanced systems were developed to improve reliability and reduce cost. As according to [7], systems such as Rotary Steerable System enabled to drill wells with complex geometries (horizontal and extended-reach wells), which were previously deemed as risky or too expensive.

2.3 Using CCTV

CCTV is increasingly used in the Oil and Gas industry as an inexpensive source of valuable data. It is mainly used for safety and security purposes, but more and more companies are extracting technical information from this type of data.

According to [10], CCTV are used focusing on security and not safety, which is a domain where there is still room for improvement. The authors emphasise the need of improving CCTV guidelines and standards supporting Human Factors.

2.4 Computer Vision methods

Human detection and human tracking are among the most useful information to extract from CCTV data for safety and security purposes. Multiple methods exist in order to do these detection and tracking and usually, a compromise between speed and accuracy will be made depending on the application.

Regional Proposal Network (RPN) based approaches
The review from [6] gives a reason why the RPN-based approaches, which are "two-shot approaches", were developed and gives a description of the different methods.

In Computer Vision, the objects of interest are both classified and located by drawing bounding boxes. As the number of occurrence of these objects varies depending on the image, it is not possible to use a standard convolutional neural network (CNN) followed by a fully connected layer. To solve this problem, the RPN-based approaches were developed: by doing a selective search to extract a limited amount of regions from the input image, with those regions called region proposals, it is possible to classify the regions with a relatively reasonable computational time (compared to the naive selection of regions). This leads to the R-CNN that will be further improved to Fast R-CNN and Faster R-CNN. As a note, Faster R-CNN is different from the previous two approaches as it uses a separate network to predict the region proposals.

You Only Look Once (YOLO) approach
The YOLO article [8] introduced YOLO as a new approach to object detection. Compared to previous approaches which use classifiers for object detection, this approach defines the problem as a regression problem. In **one** evaluation, the architecture gives bounding boxes and class probabilities. It is described as extremely fast and processes images in real-time at 45 frames per second. It does more localisation errors than other detection methods, but it is less likely to predict false positives in the background. This approach has some limitations as in particular it struggles with detecting small grouped objects. Since then, two enhanced versions of YOLO were created.

Single Shot Detector (SSD) approach
According to [13], SSD is simple compared to RPN-based approaches as it eliminates proposal generation and all computation is done in a single network. SSD is also faster than the RPN-based approaches and outperforms YOLO and Faster R-CNN in terms of accuracy (according to the results showed in the paper). Like YOLO, it doesn't perform as well as RPN-based approaches for smaller objects.

3 Code

The submitted code is written in Python. It was developed from scratch. The Agile methodology as well as a digital Kanban were used for its development.

3.1 Code metadata

Libraries and files:

- opencv
- Pytorch (with cuda) and torchvision
- pycocotools
- coco_eval.py, coco_utils.py, engine.py from cocoapi repository [1]
- utils.py from vision repository [4]
- bbox_util.py for data augmentation in object detection, from [3]

Resources used:

- Google Cloud Storage
- Dataiku (Data Science Platform)
- linux virtual environment with two K40C GPUs

Also used: LabelImg software retrieved from GitHub [11]

GitHub link to the code:
`https://github.com/msc-acse/acse-9-independent-research-project-LCS18`

3.2 Overview of code

The code is divided as such:

- Videos_to_images.ipynb

- ml.py
- custom_dataset.py
- data_augmentation.py
- compute_toto (dataiku recipe, written in Python)
- people-vs-time_plot.ipynb
- yt-loading-and-predicting.ipynb

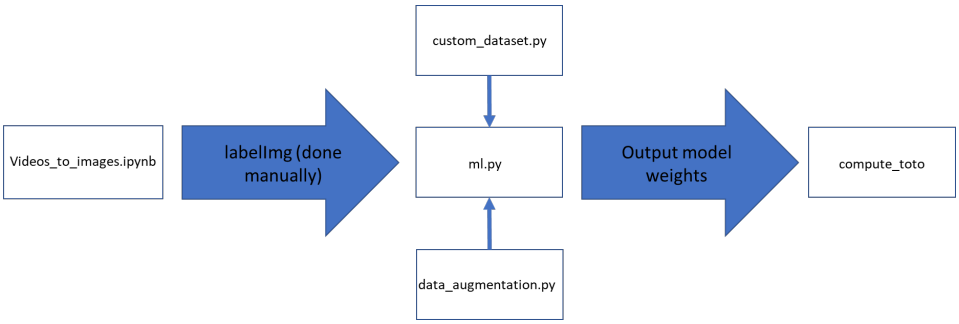

Figure 1: Code architecture

Videos_to_images.ipynb: extract frames from an input video and saves them as .jpg images in a dedicated folder also created by running this code.

ml.py: contains the main code. Loads labelled data (.jpg images and boxes from .xml files) and does the training of a model and saves it in a .pt file.

custom_dataset.py: contains the CustomImageTensorDataset class which loads the input images and their labels.

data_augmentation.py: contains the custom data augmentation classes, written for object detection, as doing data augmentation affects both the images and their labels (boxes).

compute_toto: is done on Dataiku (see figure 2 for the Dataiku flow). Loads the saved model contained in the Model_PT folder, gets the unseen videos from the RIG_VIDEOS.Videos folder and computes predictions for those videos. Those predictions are processed further to give the final results in the shape of a table (toto dataframe).

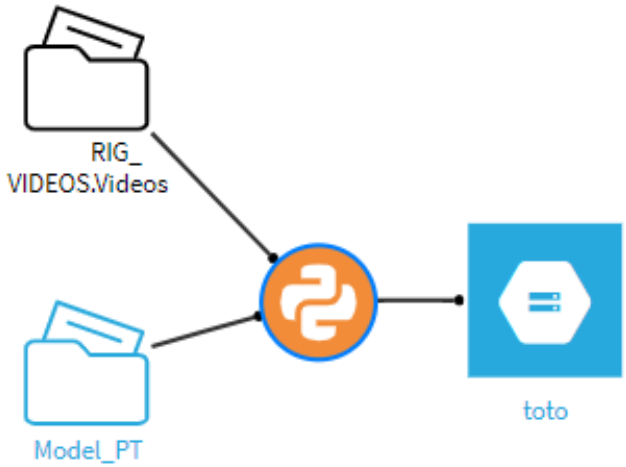

Figure 2: Dataiku flow

people-vs-time_plot.ipynb: alternative code done locally which helps computing a number of people vs time plot by counting the number of boxes detected (after "filtering" of the predictions) and getting the corresponding dates/times.

yt-loading-and-predicting.ipynb: code used to perform predictions on a youtube video and to plot the frame with its predictions.

3.3 Input and output of the main part of the code

Model training
One part of the code is dedicated to the training of a neural network for object detection. Its inputs are labelled images (images and labels stored in a dictionary). Its output is the weights of the trained model, stored and saved in a .pt file.

Evaluation using the trained model
After training the model, it is possible to do predictions with that same trained model. This requires loading the saved model from before and put as inputs a video which will be processed. The derived outputs are those predictions, which are then processed and "filtered" to deliver the final results in the shape of a dataframe. As of now, it is also possible to get the

final plot (people vs time) directly after doing the predictions: it involves a processing of those by counting the number of boxes as well as getting the corresponding times/dates.

4 Proposed approach: Object detection with Faster R-CNN + FPN architecture

This part will explain in detail the different steps taken to get to the final code.
In order to achieve the objectives of this project, object detection will be used. The following steps are taken:

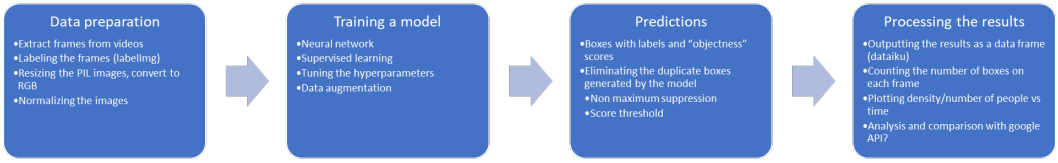

Figure 3: Main steps

4.1 Data preparation

Videos showing activities on the rig floor constitute the raw data. In order to train a model, data preparation must be done as a first step.

The model needs to be able to locate and recognise persons on the rig floor. Some of the raw data will be labelled accordingly and will constitute a dataset. The goal of this step is to construct a dataset of labelled images. To achieve this, many operations are done.

Manually resizing the data can be an important operation as it can control to some extent the amount of memory used. It is also handled internally in the chosen neural network to make sure that the neural network can process every input with different sizes. In our case, the same size will be chosen for every input image.

This operation will also have an effect on the model's performance as down-sampling will remove some specific features of the images and will allow for

the model to learn more general features.

The model was tested with some sizes of images: full size (720x1280), 4/5 size (576x1024) and 1/2 size (360x640). A better performance was obtained for the 1/2 size (the chosen architecture also turned out to accept images within this range).

Since it is intended to do supervised learning (i.e. using data with labels), labelling data will be done manually. This manual operation is also a very important step as labelling the images wrongly will lead to poor results.

Object detection involves more than just image classification. It also involves locating the objects to be classified. Thus, in this project, the data labels are boxes with their own classification, i.e. each box contains a certain object/class.

Before the labelling process, a selection of videos is made with the help of drilling data which can identify the times where an activity involving humans is taking place. As a note, different angles of camera were chosen to make the final model more generalised.

To label those data, these operations are done for a selected few:

- Extracting frames (.jpg images) from the videos and resizing those (using Videos_to_images.ipynb). They will later serve as the input images.
- Manual labelling using the labelImg software (figure 4 shows an example of application on public data).

This process was done for all the images. Producing the images from the videos and labelling them is really time consuming since not all the frames contain the object(s) of interest. Thus, labelling the data was done by multiple persons (mainly student interns) who each contributed to the final dataset. To date, the dataset is composed of 632 images, and work is ongoing to increase the size of the data, and also to evaluate the necessary size of this input dataset.

Sometimes, some difficulties could be encountered, such as recognising if the persons are on the rig floor or not (reflections on one invisible wall) or if they are even there (sometimes hidden behind objects from the background or represented by their hats or a body part).

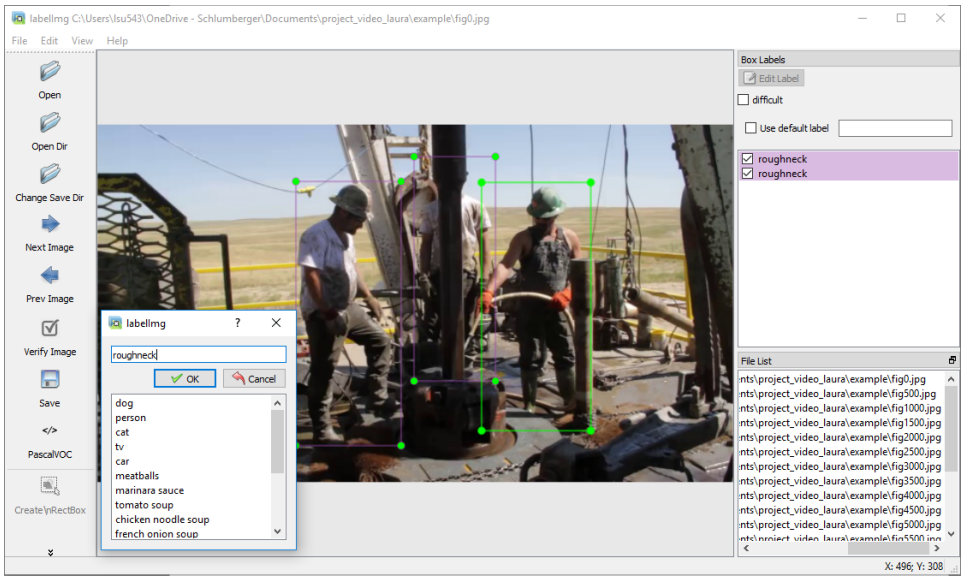

Figure 4: LabelImg software. Here, 3 people/roughnecks are at the rotary table, one of which is obscured by the drill pipe

4.2 Data pre-processing

Training, validation and test set
The whole dataset (images with labels) will be divided as training, validation and test set (figure 5).

It can be observed that the dataset is divided two times. The first splitting is done on the full dataset. The derived training set can be itself divided a second time in a "training" set and a validation set.

To obtain the final model to perform predictions and have an idea of the generalisation error, the first configuration, i.e. training set and test set will be used. The test set will serve to give an idea about the algorithm's performance on unseen data.

Concerning the training set and the validation set, this configuration will be used to tune the hyperparameters. The test set should not be used in this step as it should only serve to provide an unbiased evaluation of a final model fit on the training set. Indeed, tuning the hyperparameters using the test set will provide in the end a biased evaluation and it will not give any

true idea about the model's generalisation error.

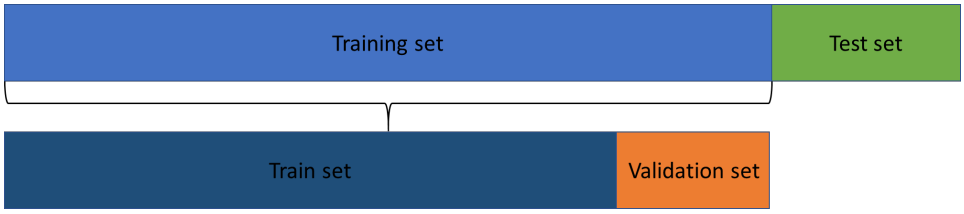

Figure 5: Train set, test set and validation set

The chosen proportions are:

- for the full dataset: training: 0.8 - test: 0.2
- for the training dataset: training: 0.8 - validation: 0.2

Those proportions can be modified to further optimise the trained model: but, with less training data, the estimated parameters of the model vary more, and with less test data, the performance statistics vary more. This could prove to be harmful to the final model.

As a note, random sampling is done when dividing the dataset.

Normalisation-standardisation
This operation is already taken care of in the actual implementation of the Faster R-CNN architecture that we will study more in detail later.

Data augmentation
In order to improve performance, data augmentation (DA) is a means to get artificially more relevant data. This will result in a more robust model.

During the training of the model only, different operations were done:

- Random horizontal flip (referred as RHF in the upcoming table), which has a probability to occur (or not)
- Random rotation, which is done with an angle randomly between -10 and 10 degrees (referred as RR in the upcoming table)

- Random scaling (crop and zoom), where the image is scaled by a factor drawn randomly between (1 - scale) and (1 + scale)

It may be observed that some data augmentation operations weren't considered, such as colour related transforms. Even though it could have help for day/night settings, it could also interfere with the "shadows" and prevent the model from achieving what we want.

The following table 1 gives a comparison of different configuration for data augmentation, with training on a dataset of 150 labelled images. All the values are proportions, i.e. between 0 and 1. Here, average precision (AP) refers to the proportion of correct predictions depending on if the ground truth box and the predicted box are relatively overlapped with each other (intersection over union), and depending on the area/size of the box. In the table, the values 0.5 and [0.5,0.95] are values of Intersection Over Union (IoU), which is area of overlap divided by area of union: the value of IoU being high means that the two studied boxes are really close to each other. The metric AP@[0.5,0.95] could be considered as mAP (mean average precision) as it is the AP for all kinds of configuration of both the ground truth box and the predicted box, with an IoU between 0.5 and 0.95, i.e. the boxes are more or less close and/or overlapping (measured by the value of IoU).

The terms "small", "medium" and "large" refer to the size/area of the box considered.

DA	AP@0.5	AP@[0.5, 0.95]	AP small	AP medium	AP large
None	0.841	0.546	0.182	0.578	0.825
RHF	0.871	0.578	0.262	0.604	0.825
RHF & RR	0.865	0.487	0.13	0.512	0.725

Table 1: Average precision depending on data augmentation

Those results show that doing a random horizontal flip seems to be the best operation to do.

Doing random rotation didn't seem to help. It may be because the videos are pretty much "straight" and their global orientation doesn't change much. However, it should be noted that these results were done with only a dataset of 150 images which were extracted from a single camera angle. It is not excluded that adding more camera angles (more diverse data) could make this operation more relevant.

As a note, even though random scaling was implemented, it was not done in practice as the random crops could give a part of the image which doesn't contain any person/box and would thus make the code crash, since the model needs both the image and its boxes (at least one box) as inputs for training.

In the future, it may be a good idea to implement a custom scaling where the relevant parts of the image (boxes) would be randomly scaled.

4.3 Neural network architecture

Now that the dataset is created, a neural network architecture must be chosen/created.

An Artificial Neural Network is a computational model inspired by the way the neural networks in the human brain process information. It consists of multiple layers of neurons: an input layer, hidden layer(s) and an output layer (figure 6, left). Each neuron receives an input and/or gives an output to the next layer's neurons and does a weighted sum of its inputs (linear). As the goal is to represent a non linear function (real life), activation functions such as ReLU or tanh can be added to introduce non linearity.

During training, back-propagation is done and can be described as "learning from mistakes": the initial model has its weights randomised and is trained. The results are compared to the desired outputs, and the error is propagated back to the previous layer, which adjusts the weights.

Compared to a simple neural network, Convolutional Neural Networks (CNN) allow for neurons arranged in 3 dimensions (figure 6, right). Some of the CNN layers are convolutional (the mathematical operation convolution on images represented as matrices is applied) and map an image to a feature map, and some other are fully connected as in regular neural networks. Pooling layers can be used to reduce the resolution of the feature map but retain features required for classification.

A classical classification with CNN is not enough as there may be multiple instances of different objects of interest in a same image where we don't know the exact number of objects beforehand. For object detection, a R-CNN architecture will be used as, in addition to classification, it is possible to get the object locations.

This work will focus on using the Faster R-CNN architecture. According

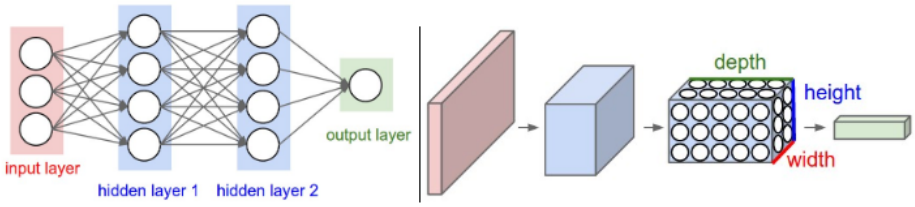

Figure 6: left: neural network, right: CNN, figure from [2]

to the literature review and what needs to be achieved, i.e. speed and/or performance, SSD or YOLO may have been better choices. Indeed, YOLO and SSD are faster, but detecting "small" objects can prove to be difficult. Those "small" objects occur quite a few time in the produced dataset. Thus, even though Faster R-CNN is deemed as slower and maybe less accurate than SSD on a particular case according to the related paper, this Faster R-CNN method was chosen as it is generally more accurate and detects small objects better than the other two methods.

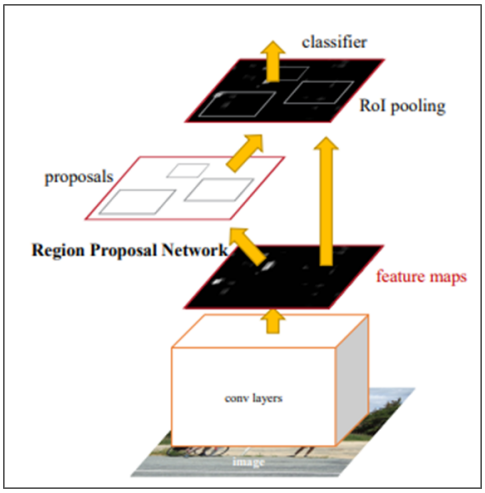

Figure 7: Faster R-CNN structure, figure from [9]

Figure 7 shows the structure of Faster R-CNN. From bottom to top of the figure: Faster R-CNN uses the same Convolutional Neural Network for both region proposal generation and object detection tasks. Instead of using a selective search algorithm on the feature map for region proposals like the previous versions of R-CNN, a separate network is used for those, as stated in the literature review. Those region proposals are then reshaped with the

RoI pooling layer. The result is then used to classify the image within the proposed regions and predict the box coordinates.

In the used implementation, the model is pre-trained and has a ResNet50 backbone and Feature Pyramid Network (FPN), which improves both accuracy and speed of the Faster R-CNN baseline. FPN replaces the feature extractor and generates multiple feature map layers with better quality information than the regular feature pyramid for object detection.

4.4 Training approach

Transfer learning
Transfer learning is one mean to save time and getting better performance. Doing transfer learning was chosen instead of training from scratch for those reasons. Indeed, the dataset must be created by labelling manually images, an operation which is really time-consuming: training from scratch requires a quite big dataset and is computationally expensive. This method will thus enable us to produce a good model with a limited amount of data.

For this project, the Faster R-CNN architecture with FPN pre-trained on COCO dataset and available on Pytorch was chosen.

Performance metrics
COCO metrics will be used to quantify the model's performance:

- Loss function: multitask loss, composed of classification loss (log loss) and regression loss (smooth L1 loss)

$$L(p_i, t_i) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i*) + \lambda \frac{1}{N_{reg}} \sum_i L_{reg}(t_i, t_i*) \qquad (1)$$

  where:

  - p_i predicted probability of anchor (box) i being an object
  - p_i^* ground truth label (1 if anchor positive, 0 if anchor negative)
  - t_i vector of 4 coordinates of the predicted bounding box
  - t_i^* ground truth box associated with a positive anchor

  - λ is a balancing parameter, and according to the paper [9], $\lambda = 10$ gives the best results
  - N_{cls} and N_{reg} are normalisation parameters

- Intersection over union:

$$IOU(A, B) = \frac{A \cap B}{A \cup B} \tag{2}$$

  - True Positive (TP): A correct detection.
    Detection with $IOU \geq threshold$
  - False Positive (FP): A wrong detection.
    Detection with $IOU < threshold$
  - False Negative (FN): A ground truth not detected.

- Precision: correct positive predictions

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

- Recall: finding all ground truth bounding boxes

$$Recall = \frac{TP}{all\ ground\ truths} = \frac{TP}{TP + FN} \tag{4}$$

Monitoring the loss and precision/recall during the training is a key step to check for underfitting (when the model doesn't fit the training data) or overfitting (when the model fits too well the training data), as both could lead to poor model performance.

The graphs (figures 8, 9 and 10) are respectively loss vs number of iterations, precision vs epoch and recall vs epoch. As the training goes on, the loss value should decrease and the curve will stabilise, which is seen on the graph. Concerning precision and recall, those values should be as high as possible. The presented graphs tend to increase during training and then stabilise to a certain value, about 0.6 for precision and about 0.65 for recall.

These behaviours show that there doesn't seem to have any overfitting. If there was overfitting, the loss value would increase again and the precision and recall would decrease too. This can happen if the model is trained for too long.

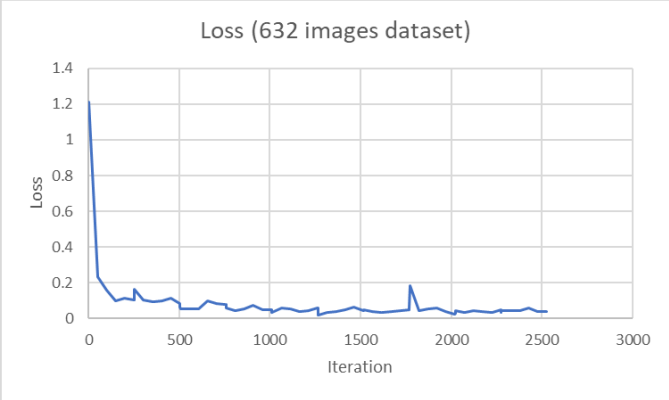

Figure 8: Loss during training (632 images dataset)

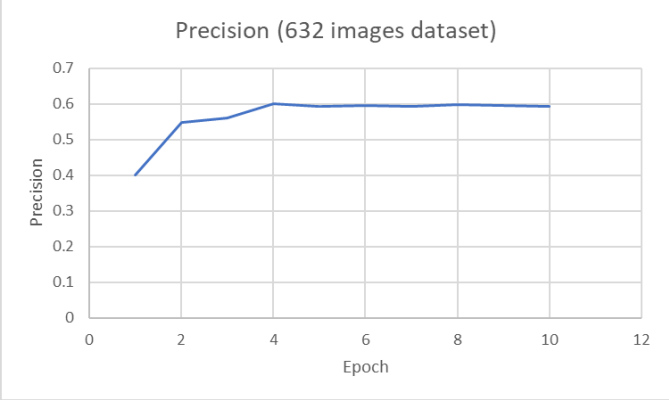

Figure 9: Precision during training (632 images dataset)

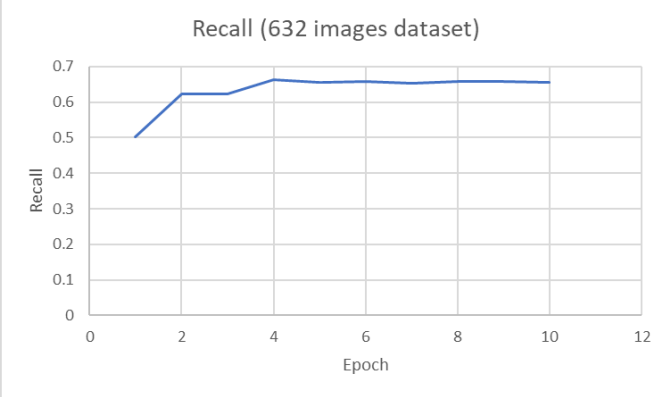

Figure 10: Recall during training (632 images dataset)

The table 2 summarises the different possible configurations and their scores. "Baseline" means here Faster R-CNN with ResNet50 and FPN, pre-trained on COCO dataset. Faster R-CNN with MobileNet pre-trained on ImageNet was also tried.

model	AP@0.5	AP@[0.5, 0.95]	AP small	AP medium	AP large
Faster R-CNN with MobileNet	0.421	0.152	0.004	0.167	0.289
Baseline - 150 images dataset	0.871	0.578	0.262	0.604	0.825
Baseline - 632 images dataset	0.898	0.594	0.114	0.616	0.693

Table 2: Average precision depending on the architecture used

The results show that using the 632 images dataset (with more camera angles) improved the model's performance for "medium sized" boxes, but not for "small" and "large" boxes. In general, the mean average precision is better when using a bigger dataset, as shown by the results.

4.5 Hyperparameter tuning

To optimise the model, a hyperparameter tuning step is done. The hyperparameters are of different origins. Both the neural network and some other helper algorithms have hyperparameters to be tuned.

For this operation, the training set can be divided further between the new training set and the validation set as described in the data pre-processing step. Performing Gridsearch is planned. However, using k-fold validation is also a possibility.

This step will be conducted during the second part of the internship. The focus to this point has been preparing the raw videos and creating a working workflow.

4.6 Prediction

After training a model, prediction can be done. As a note, data augmentation doesn't occur anymore during evaluation.
However, one problem remains: some of the resulting predicted boxes for one image may be duplicated many times, e.g. two boxes were predicted for one person. In order to solve this issue, a non-max suppression algorithm is

used. When multiple boxes are overlapping each other, this algorithm will ignore the smaller overlapping bounding boxes and return only the larger ones. The overlap threshold is the value which will decide if the bounding boxes are considered as overlapped or not.

On top of non-maximum suppression, a score threshold is also taken into account and is tuned manually depending on the results. The following figures show the effect of non-max suppression and the score threshold on the prediction done beforehand (figures 11 and 12 are examples of prediction done on public data from Youtube). To help distinguish before and after non-max suppression and score threshold, red boxes representing the results of the predictions were drawn before these operations were applied, and green boxes representing the updated results of the predictions were drawn for after the operations were applied.

Figure 11: Before non-max suppression and score threshold

In those figures, it is possible to see that some of the smaller overlapping boxes were discarded. Relatively bigger boxes were not kept after both operations because of the score threshold. It seems that after "filtering the boxes", the results are good.

In the code, this step was written after the predictions were done. This is not the best solution as all the predictions go through this process and as the predicted results need to be filtered according to this algorithm. It is possible that doing this step in the model instead would be more efficient in terms of speed as updating the results of the predictions wouldn't be

Figure 12: After non-max suppression and score threshold

necessary anymore.

5 Analysis of the results

During the project, different results were obtained using different configurations. Some of those showed us that having a proper dataset is important: indeed, some of those results involved predictions done with a model trained on 150 labelled images (only a particular type of camera angle). Also, some of the results predicted the unwanted "shadows" as persons, problem which may be solved with more data.

The latest version of the model to date is trained using a 632 images dataset.

Having these results, it is possible to plot a "Number of persons on the rig floor vs time" graph. With the confidential CCTV data, surface data (data logged during drilling operations) are also available and it is possible to compare the plot with those "measured" data. In this report, the drilling/not drilling state shown below is extracted from those "measured" data.

The results (figures 13 and 14) are an extract of the entire Well Construction operation, between 09:00:00 and 09:20:00. It is expected that during a non-drilling operation, more people would be found than during drilling operations. The results derived from object detection and the measurements seem to match each other pretty well for this example, starting at around

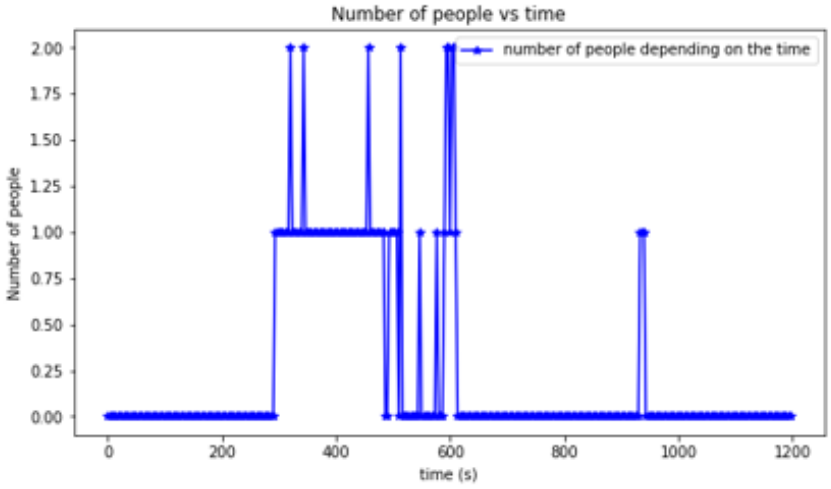

Figure 13: Number of people on the rig floor vs time during an activity, total duration of 20 min

300 seconds (5 min) and finishing at around 600 seconds (10 min) (figure 13). The measurements show about the same range of time (between 09:06:00 and 09:11:30).

As a note, it can be seen in the figure 13 that a person was detected at around 920 seconds. According to the corresponding video, one person was walking in front of the camera and left the field of view soon after. No activity was involved during the very short time when he appeared. Thus, this value was not taken into account.

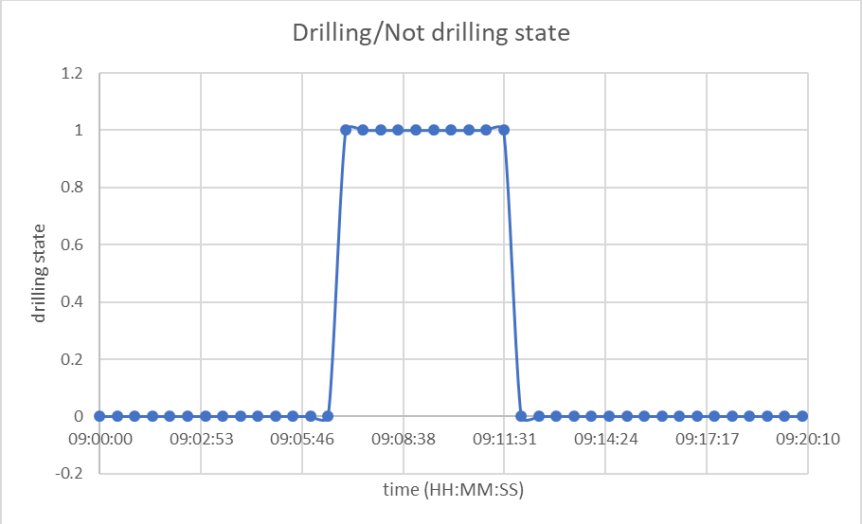

Figure 14: Drilling/not drilling state, derived from surface data

6 Conclusions

In this project, we have applied existing computer vision techniques to the problem of detecting people during Well Construction operations. The produced workflow is successfully outputting the desired results, as seen in the analysis of the results. This analysis confirmed our hypothesis of substituting the number of people for flat time. The next steps would be to tune the hyperparameters and use our model on a bigger scale (days) to cover, for example, the entire process of constructing a well. The approach could then be extended to multiple wells to assess variability in behaviour of the drilling crew.

Our solution has also some limitations. The current dataset used for training is limited as there is to date 632 labelled images. This limits the end solution which could improve by getting more labelled data.

As of now, predicting the positions of the box(es) is a pretty slow process (both training and prediction) but relatively accurate. Improving the current code and adding resources (GPUs) could potentially allow for more "real time" detection. Choosing another architecture could be needed to achieve this "real time" data processing, for applications like security and safety, but at the cost of accuracy.

Items discussed as on-going or outstanding work such as more data labelling and hyperparameter tuning will be explored further during the remainder of this 6 month internship, and reported on in a subsequent final thesis submitted to ENSG Nancy.

To conclude, the preliminary results show that the number of people could be considered as a proxy for flat time. More work will be done to expand the scale of the current analysis. Our work, which could be extended to adding a "red zone" where workers shouldn't be in for their safety, and identifying sub-activities, is only one of a wide range of application using CCTV data.

References

[1] Coco api. `https://github.com/cocodataset/cocoapi`. Works for linux, for Windows: https://github.com/philferriere/cocoapi.

[2] Convolutional neural networks (cnns/convnets). `http://cs231n.github.io/convolutional-networks/`.

[3] Data augmentation for object detection. `https://github.com/Paperspace/DataAugmentationForObjectDetection`. Git code.

[4] torchvision. `https://github.com/pytorch/vision`. Git code.

[5] Dr Paul Bommer. *A primer of oilwell drilling*. The University of Texas at Austin, 2008.

[6] Rohith Gandhi. R-cnn, fast r-cnn, faster r-cnn, yolo—object detection algorithms. `https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e`, July 2018. Accessed on 2019-06-26.

[7] Trond Skei Klausen Demos Pafitis Geoff Downton, Andy Hendricks. New directions in rotary steerable drilling. Oilfield Review, cf `https://www.slb.com/resources/publications/industry_articles/oilfield_review/2000/or2000spr02_rotary_drilling.aspx`, 2000.

[8] Ross Girshick Ali Farhadi Joseph Redmon, Santosh Divvala. You only look once: Unified, real-time object detection. *https://arxiv.org/pdf/1506.02640.pdf*, 2016.

[9] Ross Girshick Jian Sun Shaoqing Ren, Kaiming He. Faster r-cnn: Towards real-time object detection with region proposal networks. *https://arxiv.org/pdf/1506.01497.pdf*, 2016.

[10] Trine Stene Stig O. Johnsen. Use of cctv in remote operations and remote support of oil and gas fields to improve safety and resilience. *http://proceedings.dtu.dk/fedora/repository/dtu:2297/OBJ/x067.305-310.pdf*, 2014.

[11] Tzutalin. Labelimg. `https://github.com/tzutalin/labelImg`, 2015. Git code.

[12] Mike Mannering Clinton Chapman Bertrand du Castel-Randy Hansen Geoff Downton Richard Harmer Ian Falconer Fred Florence Elizabeth Godinez Zurita Claudio Nieto Rob Stauder Mario Zamora Walt Aldred, Jacques Bourque. Drilling automation. Oilfield Review, cf `https://www.slb.com/resources/publications/industry_articles/oilfield_review/2012/or2012sum02_drilling.aspx`, 2012.

[13] Dumitru Erhan Christian Szegedy Scott Reed Cheng-Yang Fu Alexander C. Berg Wei Liu, Dragomir Anguelov. Ssd: Single shot multibox detector. *https://arxiv.org/pdf/1512.02325.pdf*, 2016.