USING 3D MODELS TO GENERATE LABELS FOR PANOPTIC SEGMENTATION OF INDUSTRIAL SCENES

A. Nivaggioli [1], J. F. Hullo [1]*, G. Thibault[2]

[1] EDF Energy R&D UK Centre, BN3 5PQ Hove, United Kingdom
[2] EDF Lab Saclay, 7 boulevard Gaspard Monge, 91120 Palaiseau, France
adrien.nivaggioli@polytechnique.edu, jean-francois.hullo@edfenergy.com, guillaume.thibault@edf.fr

KEY WORDS: Deep Learning, Industrial Facility, 3D Model, Panoptic Segmentation, Labelled Dataset

ABSTRACT:

Industrial companies often require complete inventories of their infrastructure. In many cases, a better inventory leads to a direct reduction of cost and uncertainty of engineering. While large scale panoramic surveys now allow these inventories to be performed remotely and reduce time on-site, the time and money required to visually segment the many types of components on thousands of high resolution panoramas can make the process infeasible. Recent studies have shown that deep learning techniques, namely deep neural networks, can accurately perform panoptic segmentation of *things* and *stuff* and hence be used to inventory the components of a picture. In order to train those deep architectures with specific industrial equipment, not available in public datasets, our approach uses an as-built 3D model of an industrial building to procedurally generate labels. Our results show that, despite the presence of errors during the generation of the dataset, our method is able to accurately perform panoptic segmentation on images of industrial scenes. In our testing, 80% of generated labels were correctly identified (non null intersection over union, i.e. true positive) by the panoptic segmentation, with great performance levels even for difficult classes, such as reflective heat insulators. We then visually investigated the 20% of true negative, and discovered that 80% were correctly segmented, but were counted as true negative because of errors in the dataset generation. Demonstrating this level of accuracy for panoptic segmentation on industrial panoramas for inventories also offers novel perspectives for 3D laser scan processing.

1. INTRODUCTION

When engineering large industrial installations, there is a frequent need for inventories and complete understanding of the scene. For nuclear power plants for example, maintenance can be optimised if precise location of equipment is known before going on site, and deconstruction requires precise estimation of waste type and weight for planning disposal. In many cases, a better inventory leads to a direct reduction of cost and uncertainty of engineering. To also reduce the time spent on-site, a greater number of these inventories is now carried out remotely, using thousands of panoramic images captured in the plant (Hullo et al., 2015). Today done manually, the analysis of these images requires several thousand hours of tedious work for each building. Recent work in computer vision suggested that such a laborious process could be helped by an automated analysis of images taken on-site. Applications of deep learning for built environment are already numerous, from facade modelling (Schmitz and Mayer, 2016) to indoor scenes analysis (Handa et al., 2016).

When one wants to analyze, classify or segment the content of an image, deep learning techniques have proven their efficiency in the past few years. These approaches are based on systems known as deep convolutional neural networks (DCNN). In order to make accurate predictions, these neural networks need to be trained, which means that we need to have a labelled dataset of tens of thousands of examples: a set of data on which we have already performed the task we would like to automate. While online dataset are available for everyday objects, specific industrial equipment, such as those found in power plants, need a specific dataset. To avoid the high cost of manual labelling, we propose to use an existing as-built 3D model to generate a labelled dataset used for the training of image segmentation deep neural networks. We experimented our approach on quality controlled as-built 3D models of nuclear reactors.

The main body of this paper, about panoptic segmentation of indor industrial images, is split in three parts. First, we explain how we procedurally generated a large labelled dataset of panoramic images using a 3D reconstruction of an industrial scene in section 3, we describe how to prepare the dataset and review the error budget of the procedure. We then present the tools we adopted to assess the quality of our results in section 4: we detail metrics and visual tools we recommend to build an in-depth understanding of the results. Finally, we study the results of our implementation in section 5, starting with a description of the implementation of our neural networks and the associated trainings. Then, we look at some statistics detailling the quality of our panoptic segmentation. Lastly, we study the impact that the errors in the generated dataset had on our results and discuss the perspectives of this work.

2. RELATED WORK

Panoptic segmentation unifies the typically distinct tasks of semantic segmentation (assign a class label to each pixel) and instance segmentation (detect and segment each object instance) (Kirillov et al., 2018). It then covers the detection and segmentation of both the *things* and the *stuff* (Forsyth et al., 1996, Heitz and Koller, 2008, Caesar et al., 2016). A *thing* is an object that has a specific shape and dimension. It is often composed of a single object in our 3D model. If we see only one part of a *thing* object, we can easily infer what the whole object looks like. Moreover, we can count how many instances of an object is present in an image. (e.g : Lamps, Valves, Ladders). A *stuff* is an object

*Corresponding author

that has virtually no limits. It has a particular texture and global form, but can not be defined by its shape. It is often decomposed in multiple different parts in the 3D model. If we see only one part of a *stuff* object, we can not estimate where it ends. This means that we have no way of counting how many different *stuff* objects of the same class are present in an image. (e.g: Grated floors, Ventilation Pipes).

To carry out an exhaustive inventory of an industrial installation using panoptic segmentation, we then need to carry out two different tasks: a semantic segmentation of *stuff* objects and an instance segmentation of *things* objects.

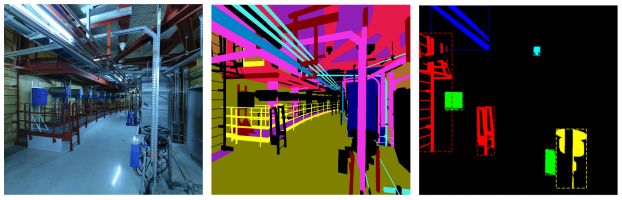

Figure 1. Panoptic segmentation task of an industrial scene. *left*: original picture. *center*: *stuff* objects, colored by class. *right*: *things* objects, colored by class.

Semantic segmentation is a pixel-wise description of the image. We attribute to each pixel the class it belongs to. Two different objects of the same class cannot be considered as two different entities. This task is has been vastly investigated in recent years (Garcia-Garcia et al., 2018). To solve this task, we used the DeepLab V2 network, that combines atrous convolution to explicitly control the resolution at which feature responses are computed within Deep Convolutional Neural Networks and atrous spatial pyramid pooling to robustly segment objects at multiple scales with filters at multiple sampling rates and effective fields-of-views, (Chen et al., 2016).

Instance segmentation consists of a pixel-wise segmentation of every *things* objects in the image. It is a combined task of detection, localisation and pixel-level contouring (Liu et al., 2018); two different objects of the same class should then be considered as two different entities. We used the Mask-RCNN network, that offers great improvements of earlier versions of RCNN, including a faster learning rate and a branch for predicting an object mask in parallel with the existing branch for bounding box recognition (He et al., 2017).

All recent DCNN are based on a deep neural network, called backbone, that usually contains tens of millions of parameters (Huang et al., 2017). The number of images required to train these network for classification ranges between tens of thousands to several millions. We used a ResNet-101 backbone with transfer learning of its 44.5 million parameters (He et al., 2016). In order to reduce the number of images needed to correctly train the networks, usually based on human and time consuming annotations, several approaches have been developed. We used a transfer learning approach in order to start with a network able to extract interesting features for regular objects, that we then trained on our cutsom classes.

To generate a dataset that would be large enough to train these deep architectures, a simple approach consists in using human made annotation. Many publicly available dataset have been created that way, but the cost of creation remains a challenge for specific applications. Other ways have been explored to reduce the cost of this stage, for example by generating a synthetic dataset using 3D renderings, such as (Peng et al., 2015), but this requires pre-existing textured 3D models. In this paper, we investigate the use of a 3D model as a mask generation tool for panoramic images of the same scene.

3. PROCEDURAL LABELS GENERATION

Many deep neural networks rely on the supervised learning paradigm: the algorithm is trained on large labelled datasets. Among the publicly available training dataset, none contains objects that can be found in a power plant. We developed and implemented a pipeline to procedurally generate image labels from a digital twin (namely a 3D model with located panoramic images), avoiding expensive manual labelling labour.

The general concept of our approach of mask generation is, for each panoramic images, to render a synthetic panorama of the 3D model using pure flat and shadows free textures ("unlit" shader), in the same position and with the same orientation as the original panorama. Each object will then be colored by a unique value encoded its class (red channel) and unique ID (yellow and blue channels).

3.1 Data source

Figure 3. Data source: as-built 3D model and panoramic image accurately positioned and oriented.

Our data source is a set of panoramic images and an as-built 3D model of a power plant building, as illustrated in Figure 3. Our 3D models are as-built ones, reconstructed from 3D laser scans that have been captured during the same period than the panoramic images. These 3D models obey a strict grouping and naming policy, allowing to identify the class of every component through its name. We consider 9 *things* classes and 12 *stuff* classes, listed in Table 1.

Panoramic images are captured approximately every 3 to 5 meters in rooms and corridors of the installation. Each native original panorama is 360° x 180° with approximately 30.000 x 15.000 pixels and all have been through a 2 levels procedure verification to assess the quality of their geometric and colorimetric stitching. We resample each panoramic image to a resolution of 4096

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-IV-2-W5-61-2019 | © Authors 2019. CC BY 4.0 License.

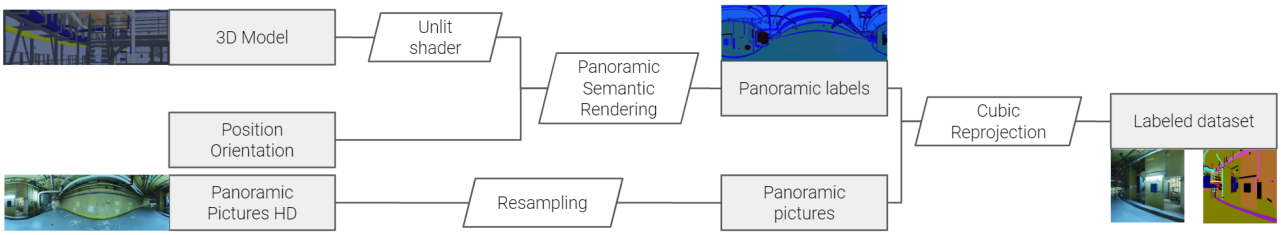

Figure 2. Overview of the procedural label generation pipeline, from a 3D model and panoramic images.

x 2048 pixels. Each panorama position and orientation are estimated in the coordinate frame of the 3D model using an interactive procedure detailed in (Hullo et al., 2015), allowing a near perfect overlay of 3D model and panoramas.

3.2 Labels extraction

Thanks to the accurate knowledge of position and orientation of panoramic images, we can use the 3D model to generate labels on a panoramic mask for each image. In order to extract these labels, we start by colouring every objects in the 3D model with an unique colour. As illustrated in Figure 4, each colour describes the class of the object and the id of the object: the Red channel is used for the class of the object, and the Green and Blue channels are used to encode the id of the object in base 256. Using this method, we can encode 256 different classes, and 65536 different objects, which was largely sufficient in our case.

Then, for each panoramic image, we generate a panoramic mask of labels using panorama's pose. We generate a panoramic rendering of the 3D model using a unlit shader that ignore normals of the surfaces and scene lights. This simple rendering encodes the object ID and gives us a complete description of the original picture where each pixel of the original image is matched to a 3D object. To avoid artifacts, we need to deactivate the anti-aliasing, which would create incoherent pixels (pixels whose value does not define the object they belong to).

Things	**Stuff**
Power box	Concrete floor
Lamp	Grated floors
Telephone	Concrete walls
Door	Cable trays
Speaker	Ventilation
Valve	Stairs
Hydrogen recombiner	Heat insulators
Fire hydrant	Pipes
Ladders	Handling structures
	Structural steelwork
	Support
	Hand-rails

Table 1. Classes of *things* and *stuff* objects from a power plant

3.3 Dataset preparation

Once we have the labels for every panoramic images, we can start using them with our neural networks. Our first tests showed that our networks did not perform well on panoramic images, the equirectangular projection adding distortions that had to be learnt by the networks (straight lines are no longer straight). We then transform each panoramic image in six 1024 x 1024 cubic images, to generate 'pinhole camera' images that the networks have originally been pre-trained on. This helps a lot our networks to learn. Also, training the networks on cubic images allows us to use them to infer on regular images, and not only panoramas, which allows us to adress a broader range of images.

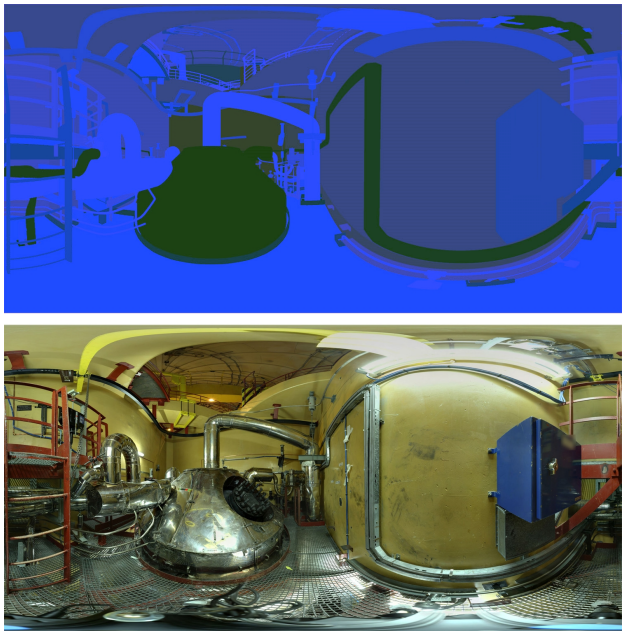

Figure 4. Label generation using an unlit rendering of a 3D model (top) where object class is encoded in the R channel and object ID in the G and B channels at the exact position and orientation of a panoramic image (bottom).

To overcome side effects of cubic reprojection, we enrich our dataset by adding transformations to the original images (and labels): we create additional cubic images with a shift of 45°. To increase the training dataset, we also add a horizontally mirrored version of each horizontal faces of the cubemaps. Finally, to overcome the limitations of current GPUs for semantic segmentation, we divided every cube face in 9 smaller images (340 x 340 pixels) to keep a sufficient resolution for the inventories of *stuff*.

3.4 Error budget

Because of the procedural generation of the labels, some errors occur and need to be investigated. An error in labeling corresponds to an incorrectly labeled pixel. All these type of errors, as illustrated in Figure 5, will negatively impact the statistical assessment of the results of the algorithm.

A first source of errors comes from the approximation of the position and orientation of the panoramas, leading to a global shift of the labels (Figure 5.a). This error has been partially mitigated with our fine registration procedure of panoramas. A second source of error comes from the geometry of the panorama itself,

reconstructed from pictures, that leads to an imperfect overlay of parts of the labels. Thanks to the quality control process of our panorama generation, this error has a low impact on our resized panoramas. The two first sources of errors will mainly impact thin objects such as narrow pipes or ladder rungs. A third source of errors comes from the level of detail of the 3D model, leading to non pixel level labels for complex objects approximated in the 3D model, typically impacting thin objects such as valve handwheels, reconstructed as plain cylinders (Figure 5.b). The fourth error comes from differences between the 3D model and the panoramas and might affect whole objects. These errors might occur when the 3D model is coarse, when the position of some objects might have changed between the 3D scanning used for reconstruction and the panoramic survey, but mostly because of temporary objects not reconstructed, such as scaffolds (Figure 5.c and .d).

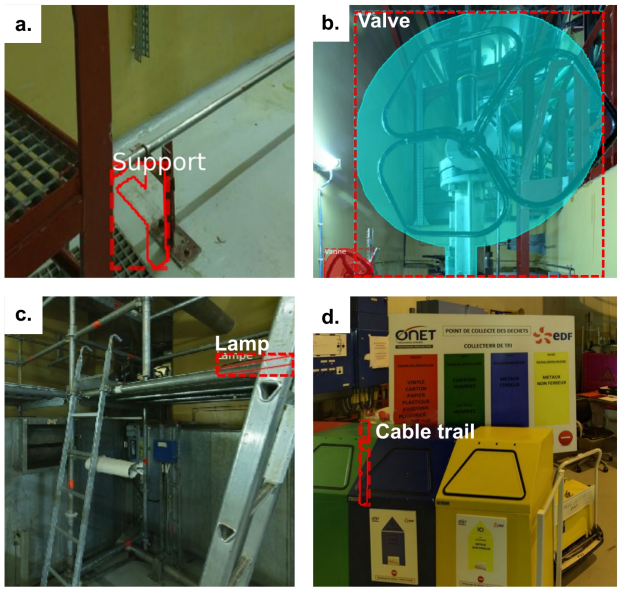

Figure 5. Errors in the labelled dataset. a) shift b) 3D model level of detail. c and d) temporary objects.

4. VALIDATION METHODOLOGY

For algorithms based on the learning paradigm, an in depth analysis of their results is crucial to evaluate their strengths and weaknesses, and hence plan their deployment. A first appreciation of results can be done through the random visual exploration of part of the results. But to explore the whole dataset, we need meaningful metrics. To understand and explore the resulting statistics, we also developed web dashboards to make the whole data analysis interactive and visual, both for inter and intra class investigations. These tools are important to explore results of both *things* and *stuff* objects. Finally, we visually investigated all remaining errors, to differentiate genuine inference errors against errors in the generated labelled dataset.

4.1 Outputs

In order to investigate the results for each types of objects, we have to compare expected and predicted outputs. Inference of *things* objects using the Mask R-CNN network generates for each instance of an object in an image a bounding box, an instance segmentation mask, a class the object is predicted to belong to, and a confidence score. Inference of *stuff* using the DeepLabV2 network generates for each pixel of an image a scoring vector whose length is equal to the number of classes (including background). The predicted pixel's class is chosen as the number of the class with maximum score. The predicted *stuff* image mask is a grid of equal resolution as the original image where each pixel value is equal to the predicted pixel's class number. Both *things* and *stuff* inference layers are merged in Figure 12 to illustrate the results if the panoptic segmentation.

4.2 Metrics

Loss functions - The first number that comes out the training stage is the loss function of the algorithm. For *things* objects, the loss function is made of 3 components that represent the losses of classification, position and segmentation as defined in (He et al., 2017). The loss function of *stuff* objects is the sum of cross-entropy terms for each spatial position in the CNN output map from (Chen et al., 2016).

Intersection over Union - Given two shapes, either a pixel mask or a bounding box, we evaluate the Intersection over Union (IoU) of expectation (ground truth) vs predicted (inference), given by the ratio of the area of intersection over the area of union. This measure gives a good and strict definition of the similarity of two shapes. For *things* objects, we can compute the IoU for either the instance mask or the bounding box; the mask IoU being more sensitive to shift errors in the ground truth dataset. For *stuff* objects, the IoU is computed for each class, using the predicted class image mask vs ground truth class image mask.

Precision and recall - In order to extend this analysis to a whole set of shapes, we define precision and recall through a confusion table (True/False Positive/Negative) for both types of objects. For *things* objects, we compute confusion table on objects instances, with thresholds on the IoU of shapes and the confidence score of the prediction. For *stuff*, we compute precision and recall for each image, at a pixel level.

4.3 Validation and investigation tools

To quantify *inter class* results, we use the class confusion matrices. They allow us to see whether the two networks mistake an object for another. We compute a different confusion matrix for each network (cf. Figure 9). The confusion matrix for *things* objects only takes into consideration how the Mask R-CNN differentiates the classes between them. For every ground truth object, it takes the prediction with the maximum IoU and check to which class it belongs. It does not take into account the objects that were not found, or the predictions that did not correspond to any objects. In the confusion matrix for *stuff* objects, the value of a cell $c_{i,j}$ represents the ratio of the pixels of class i that were predicted as a class j.

We use a second visual tool to investigate *intra class* results: IoU contour plots (cf. Figures 10, 11). For the *things* objects, we inspect how our instance segmentation network performs on each class by checking the distribution of object instances in the (IoU, Confidence) contour plot. For the *stuff* objects, we compare how well our semantic segmentation network detects objects of a specific class by plotting, for each object in the ground truth dataset, the IoU of the prediction against the size of the object in the dataset.

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-IV-2-W5-61-2019 | © Authors 2019. CC BY 4.0 License.

5. RESULTS

5.1 Implementation of panoptic segmentation

The whole label generation pipeline and training of neural networks was implemented on a gaming laptop with an Intel(R) Core(TM) i7-7820HK CPU @ 2.90GHz and a NVIDIA GeForce GTX 1070 GPU. Panoptic segmentation was implemented using two state of the art neural networks:

- Instance segmentation of *things* objects: we used the Matterport implementation of the Mask-RCNN, (He et al., 2017, Abdulla, 2017). We used a ResNet-101 backbone pretrained on the COCO dataset, and finetuned it on our own dataset, (He et al., 2016, Caesar et al., 2016).
- Semantic segmentation of *stuff* objects: we used a TensorFlow implementation of DeepLab(v2) to train a fully convolutional variant of ResNet-101 with atrous (dilated) convolutions, atrous spatial pyramid pooling and multi-scale inputs, prestrained on PASCAL-VOC dataset (Chen et al., 2016, Nekrasov, 2017).

With 1255 panoramic images describing our industrial structure, we generated 20080 training examples for the Mask R-CNN and 90360 training examples for DeepLab V2 and filtered all small objects from the training set (below 1000 pixels). This whole process took 55 hours. Training the Mask-RCNN with ResNet-101 took 61 hours; an example of result is given in Figure 6. Training the DeepLab with ResNet-101 took 42 hours; an example of result is given in Figure 7. The dataset was split in a training set (70% of images) and in a validation set (30% of the images).

Figure 6. Two examples of *things* objects predicted by the Mask-RCNN. Class colors legend is given in appendix.

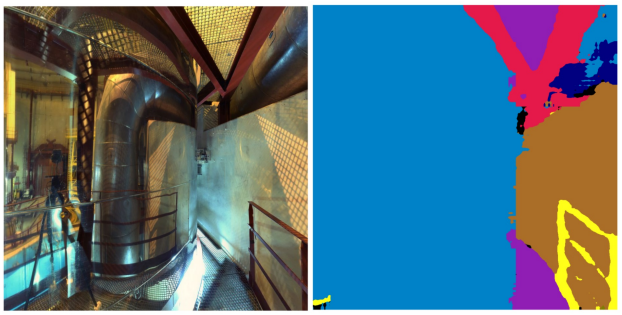

Figure 7. Example of *stuff* objects predicted by the DeepLabv2. *Left*: original image. *Right*: Masks prediction. The large blue object, a "mirror like" heat insulator, is correctly labeled. A full class colors legend is given in appendix.

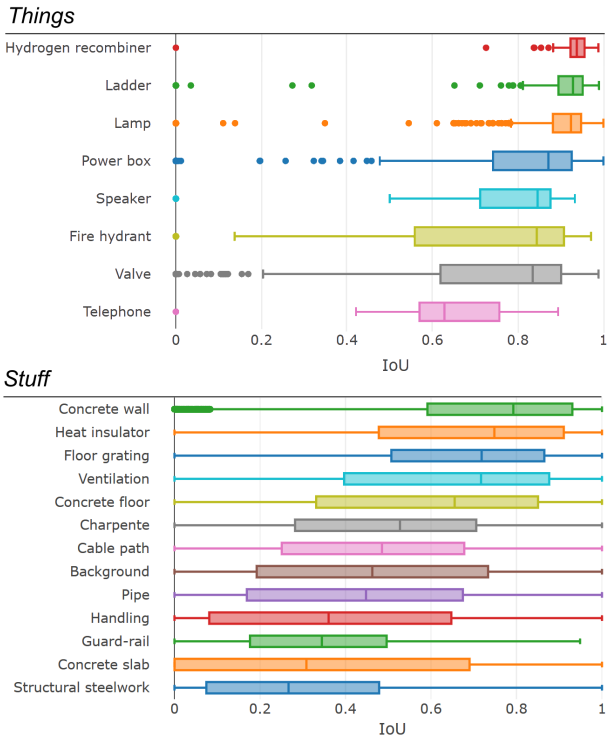

Figure 8. IoU results on the validation dataset (30 % of images). Top: *things* objects. Bottom: *stuff* objects

5.2 Quantitative results

We present here some example of statistics used for the assesment of the performances of our panoptic segmentation pipeline. Figure 8 illustrates the performances of both parts of the panoptic segmentation. The Mask-RCNN implementation delivers great results on *things* objects it has seen enough times in the training dataset. DeepLabV2 also offers great results, especially for large objects. Having in mind that the purpose of this panoptic segmentation is mainly inventories, even a low IoU offer a great feedback on the presence of objects.

Figure 9. Confusion Matrix of *stuff* objects

The confusion matrix in Figure 9 shows that thin objects like cable trays, pipes, support, handling or guard-rail are sometimes classified as concrete walls in front of which they are placed. This

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-IV-2-W5-61-2019 | © Authors 2019. CC BY 4.0 License.

can happen for two reasons: either the objects are too thin or too small to be detected, or the labels we generated are wrong. Indeed, those particular objects have a higher chance of having a wrong ground-truth: because they are thin or small, even a slight misalignment in the 3D model can make the ground truth of the object entirely false, even if the prediction is correct.

For the *things* object classes, we can inspect how our instance segmentation network performs on each class by checking how confident the network is on each prediction, and the IoU between the ground truth and the prediction. This gives us a 2D plot, on which we can calculate a density, cf. Figure 10. On the density plot of lamps, we see that the network is pretty confident and accurate when finding lamps, and that he does not make many false predictions. The density plot of ladders in the bottom of Figure 10 shows that when the network is confident in its prediction of ladders, it is mostly correct. But he predicts some wrong ladders (IoU = 0) with a lower confidence. This is not necessarily a bad thing, as we can filter our results by confidence score.

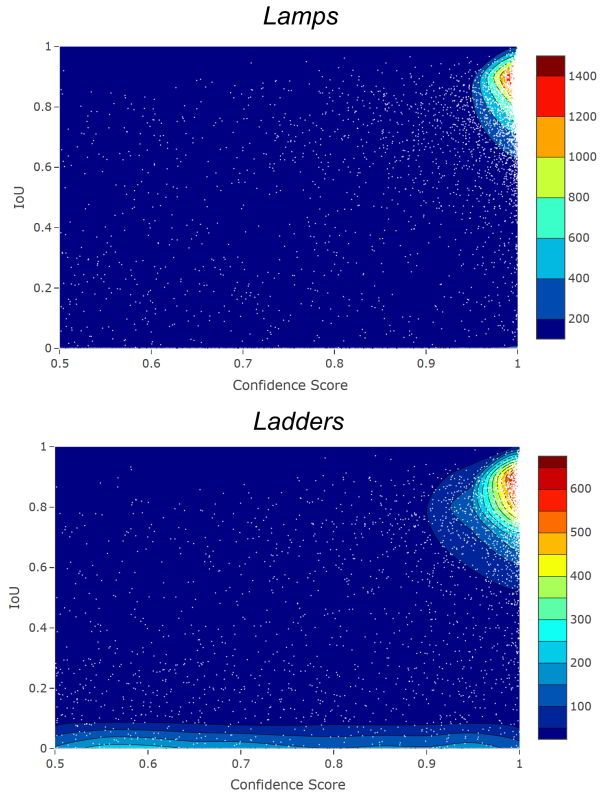

Figure 10. 2D density plot of *things* objects IoU and confidence, colored on a Jet palette; each white dot represent a predicted instance. Top: Lamps. Bottom: Ladders

For the *stuff* objects, we can compare how well our semantic segmentation network detects specific classes by plotting, for each object in the generated dataset, the IoU of the prediction against the size of the object in the dataset, cf. Figure 11. We can also define the minimum size (how many pixels) an object need to have in an image to be considered as "findable", here set at 5,000 pixels (4% of an image). This threshold discards from the box plots some thin objects that might be correctly segmented (narrow pipes), but for which the labels might not be accurate enough, due to a slight shift in the orientation typically. The top part of Figure 11 is the density plot of cable trays. This density plot seems to indicate that our network has some issues finding many of those cable trays. This is actually because cable trays are mostly quite small and in the background: their 3D reconstruction is often coarse, being of less importance for maintenance and logistics. As already illustrated in Figure 7, the density plot at the bottom of Figure 11 confirms that our network performs surprisingly really well on heat insulators. When it does not find the heat insulators, it is not because of their very specular surface, but because they only appear on a small section of the image. It is one of the major findings of this work for our industrial scenes: DeepLabv2, trained on our dataset, does not get confused by reflections of heat insulators, probably thanks to the slight ripples on their surface.

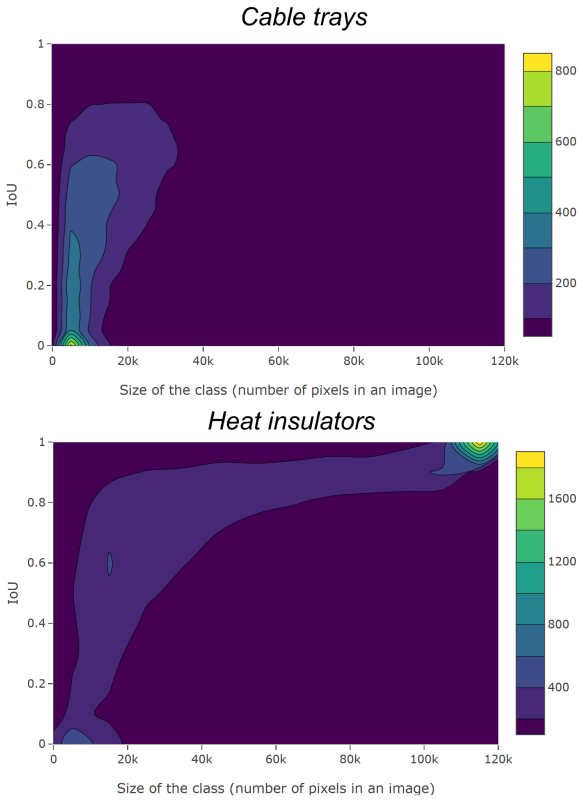

Figure 11. 2D density plot of *stuff* objects. Top: Heat Insulators. Bottom: Cable Trays

5.3 Dataset accuracy: a visual investigation

We visually investigated all 3D objects of the dataset that were not found by our networks in any panoramic image (the 20% of true negatives with an IoU of zero). We exported for each of these objects the image where the object instance was the largest.

Among the 381 3D *things* objects with a null IoU, we found that only 38% were genuine true negatives, all the others were classified as hidden, shifted, rare or nearly invisible objects. Among these true negative, 82% were small valves.

Among the 1383 3D *stuff* objects with a null IoU, we found that only 36% were genuine true negatives, all the others were classified as hidden, shifted, rare or nearly invisible objects. Among these true negative, 83% were small supports that could maybe be better found by Mask-RCNN.

6. CONCLUSION

Panoptic segmentation on panoramic images offers many opportunities for companies interested in inventories of industrial in-

stallations. It also provides a new way to control as-built 3D reconstruction. Our novel approach consists in a procedural generation of a labelled dataset used to train two neural networks (instance and semantic segmentation), as an economic way to make use of existing 3D model and located panoramic images. Despite residual errors in the training dataset, this automated process led to really good performances using state of the art neural network architectures and have demonstrated value for many generic industrial components (ladders, cable trays, valves, pipes, etc.). For some classes of objects, manually adjusting object masks is considered in order to increase accuracy of the training dataset, and hence the quality of the results.

In order to improve the accuracy of semantic segmentation, high-end GPUs would allow higher resolution inputs and implementing post-processing could also deliver more precise predictions. But beyond current panoramic images, different types of inputs are now investigated to extend the use of panoptic segmentation beyond image based inventories. Panoptic segmentation on 3D laser scanner data would lead to great advances in automation of large scale industrial engineering. The perspectives for 3D laser scanning offered by this work include the use of new information layers:

- *Depth*: RGB-Depth images, which are like regular images but with an additional depth channel. This offers more information for the networks, which could use it to better understand the property of the objects, as proposed in (Qi et al., 2018).
- *Intensity*: 3D point clouds captured with terrestrial laser scanners usually have another channel: laser intensity. This layer could give more information on the surface of an object to perform segmentation, and might even also be used as main source of information when RGB channels are not captured.

REFERENCES

Abdulla, W., 2017. Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow. `https://github.com/matterport/Mask_RCNN`.

Caesar, H., Uijlings, J. R. R. and Ferrari, V., 2016. COCO-Stuff: Thing and Stuff Classes in Context. CoRR, http://arxiv.org/abs/1612.03716.

Chen, L., Papandreou, G., Kokkinos, I., Murphy, K. and Yuille, A. L., 2016. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. CoRR, http://arxiv.org/abs/1606.00915.

Forsyth, D. A., Malik, J., Fleck, M. M., Greenspan, H., Leung, T., Belongie, S., Carson, C. and Bregler, C., 1996. Finding pictures of objects in large collections of images. In: J. Ponce, A. Zisserman and M. Hebert (eds), *Object Representation in Computer Vision II*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 335–360.

Garcia-Garcia, A., Orts-Escolano, S., Oprea, S., Villena-Martinez, V., Martinez-Gonzalez, P. and Garcia-Rodriguez, J., 2018. A survey on deep learning techniques for image and video semantic segmentation. *Applied Soft Computing* 70, pp. 41 – 65.

Handa, A., Patraucean, V., Badrinarayanan, V., Stent, S. and Cipolla, R., 2016. Understanding realworld indoor scenes with synthetic data. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* pp. 4077–4085.

He, K., Gkioxari, G., Dollár, P. and Girshick, R. B., 2017. Mask R-CNN. *2017 IEEE International Conference on Computer Vision (ICCV)* pp. 2980–2988.

He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep Residual Learning for Image Recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778.

Heitz, G. and Koller, D., 2008. Learning spatial context: Using stuff to find things. In: D. Forsyth, P. Torr and A. Zisserman (eds), *Computer Vision – ECCV 2008*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 30–43.

Huang, G., Liu, Z., Van Der Maaten, L. and Weinberger, K. Q., 2017. Densely connected convolutional networks. In: *CVPR*, Vol. 1number 2, p. 3.

Hullo, J.-F., Thibault, G., Boucheny, C., Dory, F. and Mas, A., 2015. Multi-Sensor As-Built Models of Complex Industrial Architectures. *Remote Sensing* 7(12), pp. 16339–16362.

Kirillov, A., He, K., Girshick, R. B., Rother, C. and Dollár, P., 2018. Panoptic segmentation. CoRR, http://arxiv.org/abs/1801.00868.

Liu, L., Ouyang, W., Wang, X., Fieguth, P. W., Chen, J., Liu, X. and Pietikäinen, M., 2018. Deep Learning for Generic Object Detection: A Survey. CoRR, http://arxiv.org/abs/1809.02165.

Nekrasov, V., 2017. DeepLab-ResNet-TensorFlow. `https://github.com/DrSleep/tensorflow-deeplab-resnet`.

Peng, X., Sun, B., Ali, K. and Saenko, K., 2015. Learning deep object detectors from 3d models. In: *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, IEEE Computer Society, Washington, DC, USA, pp. 1278–1286.

Qi, C. R., Liu, W., Wu, C., Su, H. and Guibas, L. J., 2018. Frustum pointnets for 3d object detection from rgb-d data. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Schmitz, M. and Mayer, H., 2016. a Convolutional Network for Semantic Facade Segmentation and Interpretation. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* pp. 709–715.

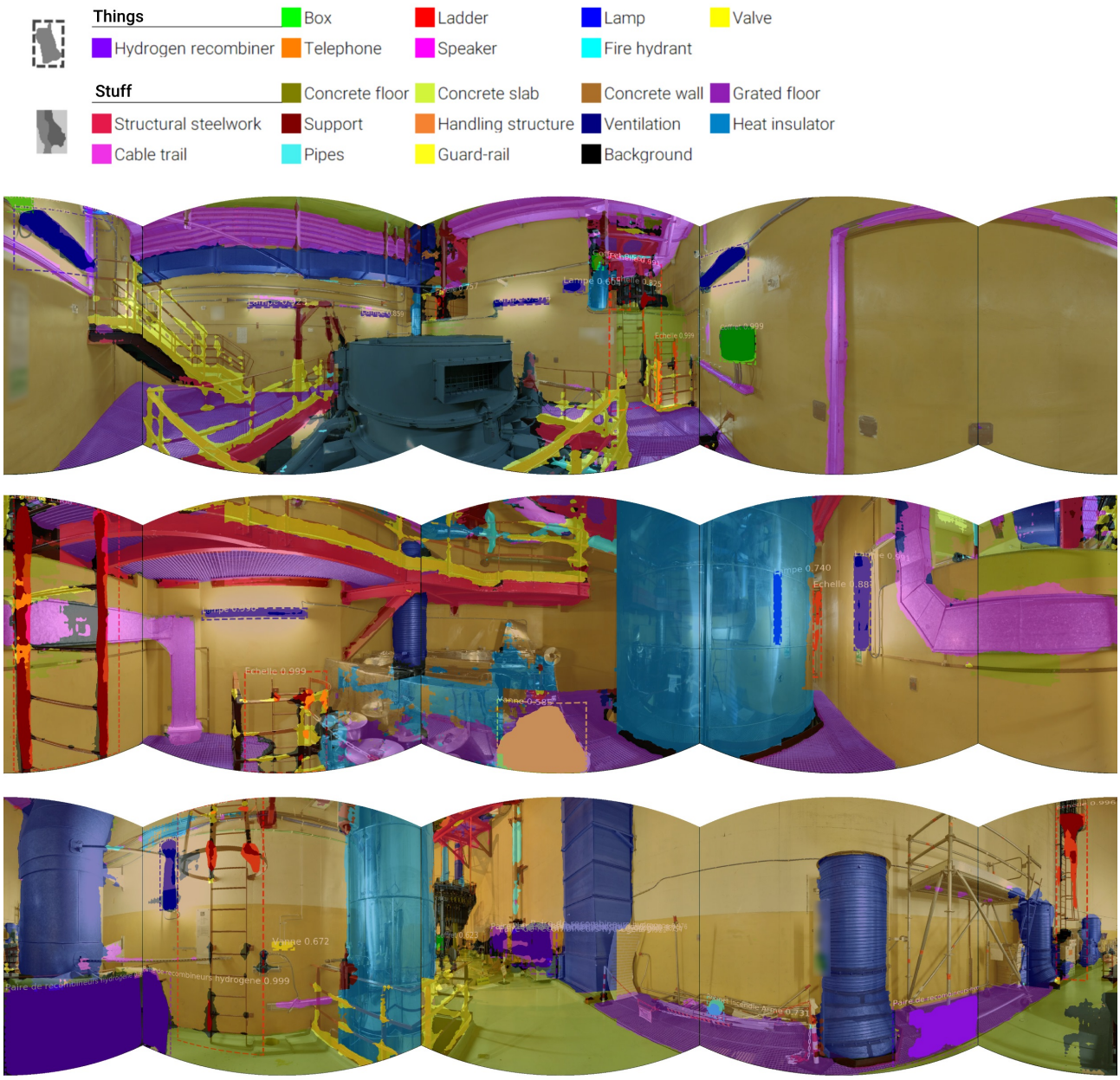

Figure 12. Examples of panoramas with predictions of *things* and *stuff* objects.