DOI: 10.23784/HN119-03

Automatic boulder identification in side-scan sonar

An article by JESPER HAAHR CHRISTENSEN

Boulder surveys seek to identify prominent boulders which position may collide with planned cable routes, offshore wind farms or other subsea construction activities. Data is collected using suitable sensor technologies such as bathymetry from multibeam echo sounders and side-scan sonar imaging. Currently, the boulder identification process is a labour-intensive job that requires domain expertise to interpret the data and provide each identified target with accurate annotations. With this work, we propose to automate the majority of this process by training neural networks to identify boulders in side-scan data. Our preliminary work estimates the area covered by each boulder instance and further generates metadata for each identified target for filtering, sorting and report generation. In addition to being an automated process, our method can process several kilometres of side-scan data and identify thousands of boulders in less than a minute. Not only does this provide results of high accuracy but it also performs orders of magnitude faster than human processors.

boulder identification | AUV | SeaCat | artificial intelligence | deep learning | side-scan sonar imaging
Erkennung von Felsbrocken | AUV | SeaCat | künstliche Intelligenz | Deep Learning | Side-Scan-Sonar-Bilder

Bei der Vermessung von Felsblöcken geht es darum, markante Felsblöcke zu identifizieren, deren Position mit geplanten Kabeltrassen, Offshore-Windparks oder anderen Bauten unter Wasser kollidieren könnte. Die Daten werden mit Hilfe geeigneter Sensortechnologien gesammelt, wie z. B. Bathymetrie von Fächerecholoten und Side-Scan-Sonar-Bildern. Derzeit ist es eine arbeitsintensive Aufgabe, Felsbrocken zu erkennen, die Fachwissen erfordert, um die Daten zu interpretieren und jedes identifizierte Ziel mit genauen Anmerkungen zu versehen. Mit dieser Arbeit schlagen wir vor, den Großteil dieses Prozesses zu automatisieren, indem wir neuronale Netze trainieren, um Felsbrocken in Side-Scan-Daten zu identifizieren. Unsere vorläufige Arbeit schätzt die Fläche, die von jedem Felsbrocken abgedeckt wird, und generiert darüber hinaus Metadaten für jedes identifizierte Ziel zum Filtern, Sortieren und Erstellen von Berichten. Da es sich um einen automatisierten Prozess handelt, kann unsere Methode in weniger als einer Minute mehrere Kilometer an Side-Scan-Daten verarbeiten und Tausende von Felsbrocken identifizieren. Dies liefert nicht nur Ergebnisse von hoher Genauigkeit, sondern arbeitet auch um Größenordnungen schneller als Menschen.

Author
Jesper Haahr Christensen is Ph.D. student and works at Atlas Maridan ApS in Denmark, a subsidiary of Atlas Elektronik GmbH, Germany.

jhc@atlasmaridan.com

1 Introduction

Collecting side-scan data of large areas using autonomous underwater vehicles (AUVs), operators seek to utilise the vehicle's total endurance to cover as much area as possible by following a predefined trajectory or by traversing back and forth in a lawnmower pattern. This is accomplished with a carefully chosen set of parameters relating to vehicle trajectory, attitude, velocity and sensor settings to align for sufficient coverage, overlap, data resolution and data quality. As work is done in GNSS-denied environments, positioning and navigation are estimated by high-quality acoustically-aided subsea inertial navigation systems. During operation, communication with the vehicle is limited to low-bandwidth acoustic data links typically used to monitor only a few important vehicle parameters.

We describe subsea surveying as the process of mapping the ocean floor by collecting highly accurate georeferenced data points. Boulder surveys, more specifically, seek to identify prominent boulders whose positions may collide with planned cable routes or offshore wind farms. Such surveys are performed using suitable sensor technologies such as bathymetry from multibeam echo sounders and side-scan sonar imaging. The ever-accumulating log files are first available for review after mission completion where boulders, defined as salient objects on the seabed, need identification. This identification process is a labour-intensive job and does not only require domain expertise for interpreting the data, but each identified target needs further to be measured in size and mapped to its exact location. The processing job is typically concluded by compiling a report of all prominent targets and their metadata, providing the client with detailed information needed for making informed decisions on the next steps.

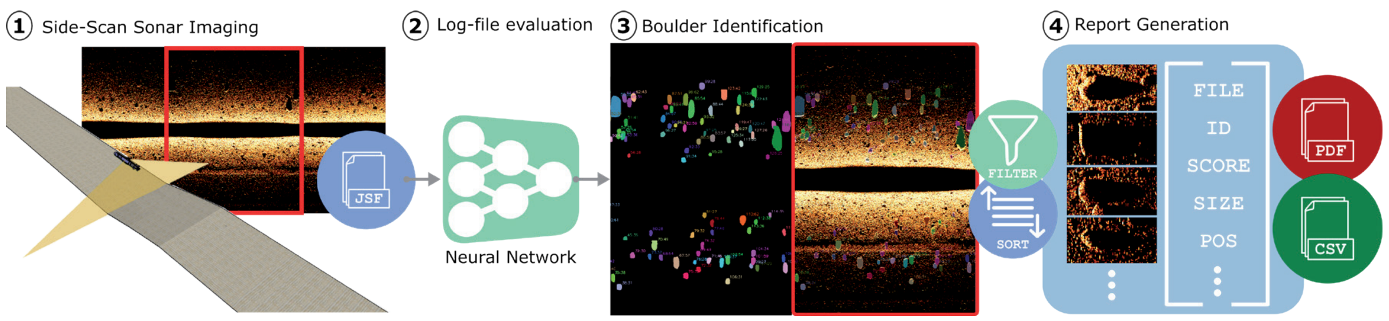

Fig. 1: Boulder identification in side-scan sonar data. 1. Underwater vehicle on boulder survey collects and logs side-scan data. 2. Side-scan logs are evaluated by a neural network that predicts plausible areas occupied by a prominent boulder. 3. Boulders are identified and metadata is computed, filtered and sorted. 4. Side-scan logs are concluded by generating a report. Colour map is applied for better visual appearance

As shown in Fig. 1, we propose to automate the majority of this process by leveraging neural networks for processing side-scan logs. More specifically, recorded logs are passed through our model that estimates the positions and areas of prominent boulders. Due to the design nature of the neural network, all estimated boulders are separable at an instance level. This allows us to count the number of boulders in the log file and analyse each separate boulder in relation to georeferenced position, size and confidence. By further filtering the outputs, we can remove most false positives and eliminate cases where a boulder is identified and counted several times. The latter case occurs due to overlap in the data, which typically is created and desired by design. We currently make a naive assumption on size and calculate a pseudo area for each boulder using the coverage obtained by the estimated segmentation masks and can thus sort boulders by size. Future work will be making more accurate predictions on boulder sizes and hence give complete control over which sizes (width, height and length) are desired to report.

Our method does not only benefit from being an automated process but also promises substantially shorter processing times. Quantitatively, our model can process several kilometres of side-scan data and identify thousands of boulders in less than a minute. In addition, the number of boulders that our model identifies is significantly higher than any human processor would possibly have time to process. The presented work is ongoing, but we estimate that saved time and effort will be substantial even in the current state.

2 Side-scan sonar data set

We collect available side-scan logs from previous completed in-house surveys and prepare them in a format suitable for training neural networks, as detailed below.

2.1 Hardware and data details

All side-scan data is recorded using EdgeTech 2205 sonar systems integrated on our SeaCat AUVs (Kalwa 2019). Data is recorded using high- and low-frequency channels and is available in EdgeTech's JSF file format. Typically, side-scan logs are split into line segments where turns are excluded. The spatial size of each line segment varies with velocity, altitude and sensor settings. Our complete data set consists of more than 1500 km worth of line segments recorded at many different locations. However, only a tiny fraction of this has been given annotations suitable for learning boulder identification. In total, we have annotated about 1300 prominent boulders with varying settings and visual variations. In Fig. 2, we show two side-scan snippets with corresponding segmentation masks. We note that the objective of segmentation masks is to cover the complete area occupied by prominent boulders in order to, at a later stage, separate each of these areas into a boulder and shadow region.

2.2 Preprocessing of acoustic images

With the protocol description from EdgeTech, we read raw JSF files using Python to load the sonar trace data into matrices for high- and low-frequencies and port- and starboard-side channels. We further collect the navigational data, pulse information and weighting factor for each cross-track line.

With the weighting factor and additional sensor information, we restore each data sample to its original floating-point value and perform slant-range correction to match along-track and cross-track resolutions. Each sonar image is further nor-

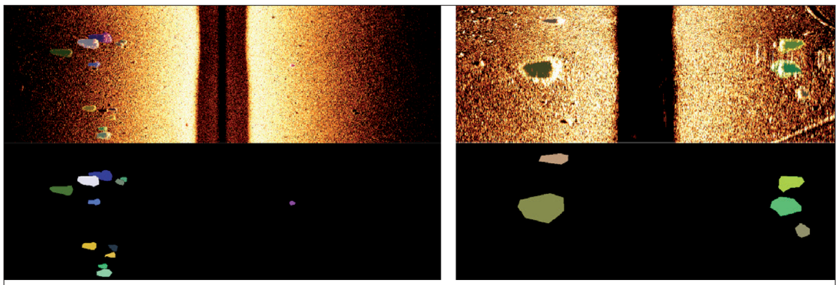

Fig. 2: Side-scan snippets with corresponding segmentation masks. For each boulder, we draw a segmentation mask that covers the boulder itself and its shadow. Colour map is applied for better visual appearance

malised and stored as PNG images split into 200 m long segments. When applying colour maps for visualisation, we use the cmapy Python module with the afm-hot colour scheme.

3 Boulder identification model

We aim to train a neural network that, given an input, predicts plausible areas occupied by boulders not only by placing rectangular boxes around the object but by a per-pixel classification of the input. Typically segmentation models consist of an encoder-decoder style architecture like U-Net (Ronneberger 2015). However, while U-Net style networks are fast and easy to train, they fail to offer object-instance separation. Consequently, segmentation masks predicted by the model are essentially one object and thus need further processing to identify and analyse each separate boulder. Especially when boulders are located in close proximity, this is a difficult task. To avoid inaccuracies from hard-coding object separation algorithms, we base our model on Mask R-CNN (He et al. 2017) for instance segmentation. As shown in Fig. 3, several neural networks are used as explained in the following.

3.1 Feature extraction

To generate feature maps of our input, we use a Feature Pyramid Network (FPN) (Lin et al. 2017) build on a ResNet (He et al. 2016) CNN with 50 layers. The FPN is used to extract features at different scales from our single-scale input efficiently. This is analogous to processing our input image at different scales but much more efficient in terms of computation. The complete ResNet-50-FPN backbone is pre-trained on the Microsoft COCO: Common Objects in Context data set (Lin et al. 2014), and is not updated during training for boulder identification.

3.2 Region proposals

Region proposals are generated by a Region Proposal Network (RPN). This evaluates the input feature map and predicts a set of rectangular regions and their objectness (score of object vs. background). The regions are generated by sliding a fixed set of windows, called anchors, with varying aspect ratios and scales over the available (differently scaled) feature maps generated by the FPN. We refer to Ren et al. (2015) for more details on RPNs.

3.3 RoI pooling

The Region of Interest (RoI) pooling layer accepts the feature map generated by our ResNet-50-FPN feature extractor and the proposals from the RPN. The proposals from the RPN are a set of regions, each defined as a four-tuple (r; c; h; w) that specifies top-left corner (r; c) and its height and width (h; w). As such, RoI pooling has the objective of »cropping« out regions of the feature map in which the RPN has estimated an object and passing it on to the final output heads.

3.4 Box and score prediction head

This accepts the feature map of the regions proposed by the RPN, i.e., the regions most likely to contain a boulder. We have two objectives for this head: (i) bounding-box regression and (ii) object classification/score. The bounding-box regression refines the region proposed by the RPN to enclose the object better. The score is a measure of the network's estimated probability of this object being a boulder.

3.5 Mask prediction head

For generating segmentation masks of our object instances, a Fully Convolutional Network (FCN) (Long et al. 2015) is used to estimate binary masks for each RoI. This prediction is 1-channel and binary, and thus class-agnostic. Therefore, we rely on the classification score from the above prediction head to identify the object correctly. As we currently focus on boulders as a whole, we only have two options for the classification (background or

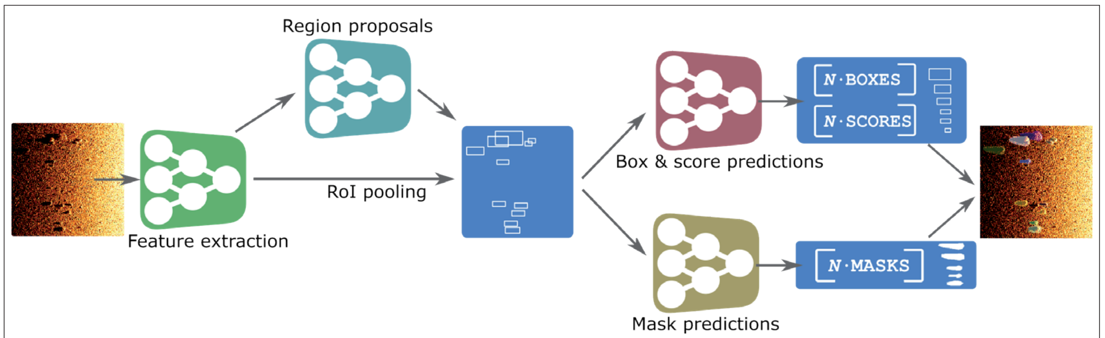

Fig. 3: Boulder identification model overview. A feature extraction network generates feature maps of the input image. These are the input to a region proposal network (RPN), predicting regions in the input likely to contain an object. Using the region proposals, a region of interest (RoI) pooling layer »crops« the feature map and passes the regions on to our prediction heads. The box and score prediction head refines the bounding-boxes for the regions and estimates the probability of each region being a boulder. The mask prediction head performs a per-pixel classification of the proposed regions and thus outputs a segmentation mask for each proposed region. Colour map is applied for better visual appearance

boulder). However, in future work, we can use this same architecture for expanding the scope to include more low-level predictions, e.g., separating boulders into boulder and shadow regions. For more detailed information on the mask prediction head, we refer to He et al. (2017).

3.6 Loss functions

As we train our entire network end-to-end, we employ a multitask loss that seeks to minimise the error in each sub-network simultaneously. We can define the total loss as:

$$L = L_{RPN} + L_{loc} + L_{cls} + L_{mask} ,$$

where $L_{RPN} = L_{loc}^{RPN} + L_{cls}^{RPN}$ constitutes bounding-box regression loss and objectness score of the RPN, L_{loc} and L_{cls} are the bounding-box regression and classification loss of the box and score prediction head, and L_{mask} is the average binary cross-entropy loss over the per-pixel classification of the RoI for the final segmentation mask (mask prediction head). For more details on loss functions and their implementation, we refer to Girshick (2015), Ren et al. (2015) and He et al. (2017).

4 Experiments

4.1 Training details

Our models are implemented in PyTorch and trained using four NVIDIA RTX 2080 Ti GPUs. During training, we load from our data set the 200-m tracks with corresponding segmentation masks and »mine« regions in which boulders are located. We do this by random cropping 256 × 256 pixel areas at locations that contain at least one annotated boulder. Since our annotations are sparse in the sense that not all boulders in a 200-m track have been carefully annotated, we ignore areas with no corresponding annotated segmentation mask. We further apply image transformations at random during training time to represent as many visually varying examples as possible. Finally, before entering the network, the input image is resized to 800 × 800 pixels. We use a batch size of eight per GPU and train our complete model end-to-end for about 5×10^3 update iterations.

4.2 Results

During inference, as illustrated in Fig. 4, we process side-scan data line-by-line by sliding a 256 × 256 pixel window over it. We note that we can process input images of arbitrary size during inference time and are not limited to non-postprocessed (raw) lines. Hence, if a post-processing step is used to generate side-scan mosaics and mitigate errors from erroneous navigation or correct position offsets due to uneven seabed, we can utilise this as input to our model. We use a sliding window approach to generate overlap of the processed data. This ensures that objects receive maximum exposure to the network and are therefore identified as a whole and not only partly identified. The stride of the window is an adjustable parameter that balances processing time and accuracy. We use a stride of 100 pixels for our experiments.

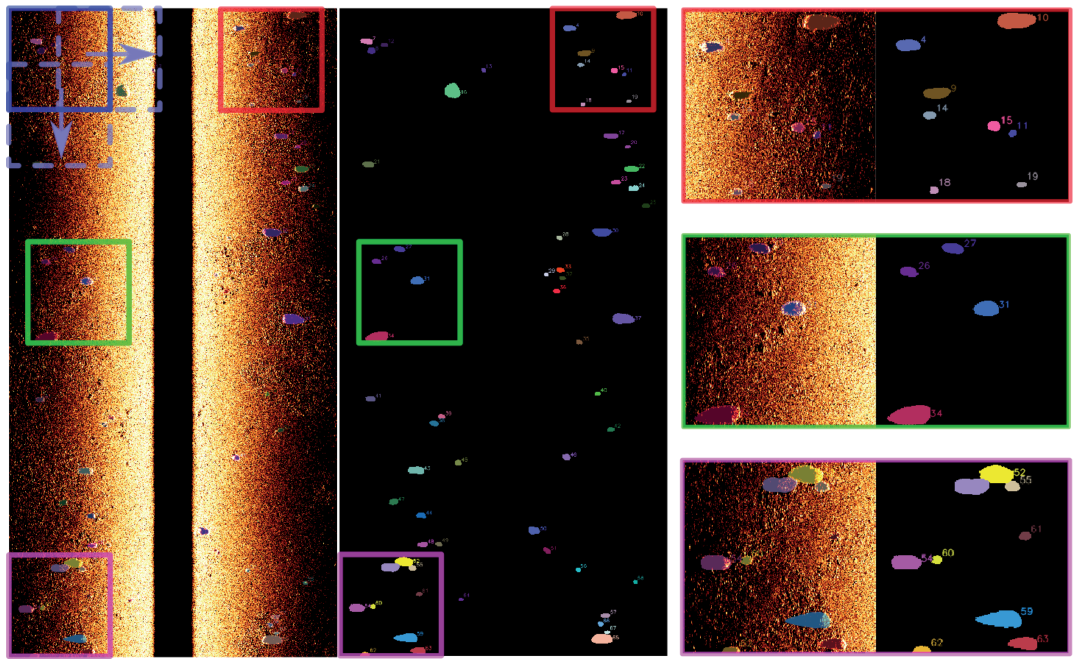

Fig. 4: Result from processing a 200-m side-scan line. Inputs to the model are generated by sliding a 256 × 256 pixel window over the data as illustrated by blue squares. The red, green and purple squares are enlarged results and shown to the right. Note that the input to the model is greyscale, but a colour map is applied for better visual appearance. The input image used here has a length of 200 m, a width of 80 m and a resolution of 0.1 m/px. This was processed on a single GPU in 13 s. The amount of identified boulders does not affect the processing time

Test data set	Non-filtered	Filtered
Accuracy (annotated vs. identified)	91.7 %	91.7 %
Accuracy (non-annotated vs. identified)	87.8 %	**94.8 %**
Model gain (annotated vs. identified)	× 62	× 27

	Human	Model (stride = 100 px)
Average processing speed	0.5 m/s	**22 m/s**
Average time per boulder	21.7 s	**0.1 s**

Table 1: Comparison of accuracy and performance on the test data set. See section 4.3 for details on the metrics presented

After processing each window separately, we reassemble outputs corresponding to the input by mapping each locally detected boulder into its global coordinate. Due to the sliding window approach, many boulders will be identified several times. To keep only the best segmentation mask for the identified boulders, we use non-maximum suppression (NMS) that filters overlapping objects based on their intersection over union (IoU) and keeps only the best scoring.

During inference, we also collect all available metadata relevant to each identified boulder. This is currently logfile name/path, id, position, confidence and »pseudo«-area. The »pseudo«-area is calculated as the area covered by the estimated segmentation mask and is for sorting purposes assumed to correlate with the actual size of the boulder. From the metadata, human operators can further filter and sort the outputs to obtain the desired output. Finally, the information can be compiled into a report, e.g., as CSV or PDF files.

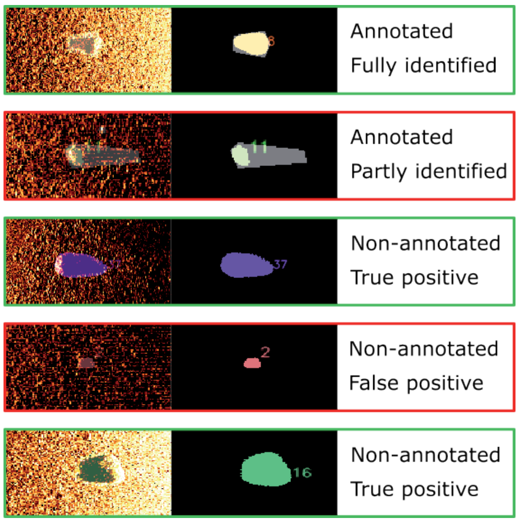

Fig. 5: Representative samples from the test data set. True positives have masks that capture the shape of targets (boulder and shadow) with high accuracy. False positives fail to capture the entire object or estimate objects at positions where there are none. Annotated masks are shown in faded white as an overlay. Each example has been resized, and sizes are therefore not relative to each other

For the input shown in Fig. 4, the time used for the complete processing steps from input to output on a single GPU is 13 s. The input has a length of 200 m, a width of 80 m and a resolution of 0.1 m/px. The amount of identified boulders does not affect the processing time.

4.3 Comparison against human-annotated data

To provide quantitative metrics on the performance of our model, we compare the targets found by our model against the human-annotated targets on our test data set. Table 1 shows the accuracy with which our model identifies targets annotated in the test data set, i.e., the proportion of annotated targets the model correctly identifies. Since our model further identifies many targets beyond the annotated targets, we also provide the accuracy of these being correct, i.e., the proportion of correctly identified non-annotated targets. The model gain denotes the gain or increase in identified targets compared to the number of annotated targets, e.g., a gain value of × 62 refers to the model identifying × 62 more targets than have been annotated. Finally, we report the processing speed as a measure of metre per second using a single GPU and time spend on generating segmentation masks using human annotators and our model. We note that these measures are an average over our test data set collected using a window stride of 100 px and are not entirely representative. They are more dependent on sensor/vehicle/data parameters than the number of boulders identified. The column »Non-filtered« denotes that all outputs from the model have been used without modification, and »Filtered« refers to the outputs being filtered. The filtering is currently removing identified boulders with an area smaller than the smallest of the annotated targets.

In Fig. 5, we show representative examples of the different cases, which is the basis for the metrics reported in Table 1. True positives, annotated or non-annotated, are shown as masks that capture targets' shapes (boulder and shadow) with high accuracy. We note that the estimated masks typically enclose the objects better than our annotated masks (shown in faded white as overlay). False positives, annotated or non-annotated, are shown as objects only partly identified or by estimated masks that do not cover any objects. We note that false positives relate mostly to small areas, further supported by the increase in accuracy of the filtered metrics as shown in Table 1.

5 Conclusion

We have presented our preliminary work on automatic boulder identification in side-scan sonar data. With our method, we currently identify prominent boulders by estimating a segmentation mask that accurately captures the entire area of the boulder and its shadow. As our model is based on instance segmentation, we retrieve and analyse each boulder separately to provide each detected target with metadata used for filtering, sorting and report generation purposes. Using only a single GPU, our model can process several kilometres of side-scan data and identify thousands of boulders in less than a minute. We envision that even in its current state, our presented work has the potential to drastically reduce the effort of industry professionals even if human-in-the-loop is still required to some extent.

6 Future work

In improving our method, several steps may be considered. To improve the accuracy, we can use our current model to label our entire data set of more than 1500 km side-scan sonar data in a semi-supervised manner. This potentially generates millions of annotated boulders to retrain on instead of the 1300 targets used for this work. To report more accurate measures on size (width, height, length), the segmentation masks may be extended to separate the currently estimated masks into a boulder and shadow area. Our model architecture readily supports this addition and thus only needs labels for learning. Finally, for accurate side-scan object positioning, we currently assume a flat seabed, and to remove duplicate identifications on separate side-scan lines with overlap, we must assume good navigation. To mitigate these assumptions, either post-processed mosaics must be used (already supported), or further work may investigate cross-referencing data points with accurate bathymetry from, e.g., multibeam echo sounders. //

Acknowledgements

We thank Fugro Germany Marine GmbH for providing support and insights in boulder identification.

References

Girshick, Ross (2015): Fast R-CNN. 2015 IEEE International Conference on Computer Vision (ICCV), DOI: 10.1109/ICCV.2015.169

He, Kaiming; Xiangyu Zhang et al. (2016): Deep residual learning for image recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), DOI: 10.1109/CVPR.2016.90

He, Kaiming; Georgia Gkioxari et al. (2017): Mask R-CNN. 2017 IEEE International Conference on Computer Vision (ICCV), DOI: 10.1109/ICCV.2017.322

Kalwa, Jörg (2019): Unter-Wasser-Drohnen für Hydrographie und Seebodenerkundung. Hydrographische Nachrichten, DOI: 10.23784/HN114-02

Lin, Tsung-Yi; Michael Maire et al. (2014): Microsoft COCO: Common Objects in Context. In: Computer Vision – ECCV 2014, DOI; 10.1007/978-3-319-10602-1_48

Lin, Tsung-Yi; Piotr Dollár et al. (2017): Feature pyramid networks for object detection. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), DOI: 10.1109/CVPR.2017.106

Long, Jonathan; Evan Shelhamer; Trevor Darrell (2015): Fully convolutional networks for semantic segmentation. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), DOI: 10.1109/CVPR.2015.7298965

Ren, Shaoqing; Kaiming He et al. (2015): Faster R-CNN: Towards real-time object detection with region proposal networks. Advances in Neural Information Processing Systems, DOI: 10.1109/TPAMI.2016.2577031

Ronneberger, Olaf; Philipp Fischer; Thomas Brox (2015): U-net: Convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention (MICCAI), Springer, http://lmb.informatik.uni-freiburg.de/Publications/2015/RFB15a