Requirements for all task implementations

- Automatically detect boats @ ~1fps
- Processed image resolution should be as high as possible, as higher resolutions allow us to fly higher and therefore observe a bigger area
- Missing boats / false negatives are not allowed as we are handling human lives
  - Better have a few false positives than a false negative
  - Preprocessing will double check detections over time by existing algorithm
  - System is usable until there is about one false positive every few minutes / 100 pictures
- Two possible target platforms for the algorithms - Try to optimize the processing needs for one of them
  - Onboard – Online @ Drone
    - Too low bandwidth (max. ~100kbit/s) for live video due to high distance to ship
    - Send cut out images of detected boats
    - Algorithm need to work on low level hardware like Raspberry Pi
  - Offboard – Offline @ Ship
    - Run algorithms on images after landing of the drone
    - Algorithms should work in reasonable time on laptop with GPU
- Evaluation
  - Use average precision implementation from dev kit

TEAM A

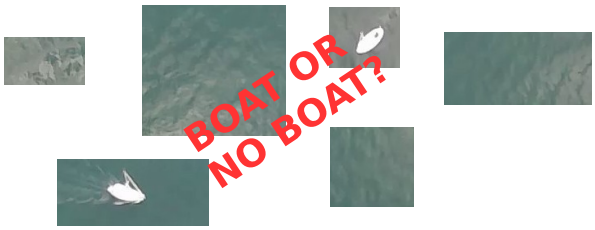
Single frame – image and object proposal classification

Needed Skills: Python, Deep Learning

Why
This task is split in two parts.
Image classification

Its already a big help, if we know if a image from the dronecamera got a boat inside or not. This can be achived by applying a image classification algorithm to the image.
After classifiying mulitple images in a row and detect consecutive frames with boats, we can apply more detailed algorithms to determine the position of the boat.

Objectproposal classification



The existing code for the boatdetector can already detect proposals / regions of interest (ROIs) in every single camera image. Only the 3d position information of these ROIs will be currently used for validation of the proposals. The ROIs could be analyzed by using a image classification algorithm to detect boats faster compared to the classic approach.

Details
Image classification
The classification should work quite straight forward by using a deep learning image classificator.

Objectproposal classification
Here you can also use a image classifier, but you need to consider that every image got different width and height. Rescaling can distort the shape of the objects in the image. One way to deal with this is to use zero-padding to fit the rescaled images to the network input dimensions. It can be also prevented that the loss calculation is influenced by the zero-padding.

State-of-the-art image classification algorithms can be found here:
https://paperswithcode.com/sota/image-classification-on-imagenet

Data
Image classification
Bodensee
Airbus
Objectproposal classification
Bodensee
Airbus
DOTA

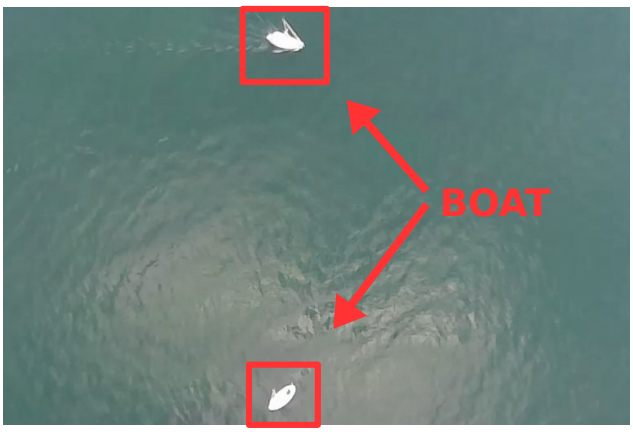
Evaluation
Average Precision

TEAM B

Single frame – object detection

Needed Skills: Python, Deep Learning

Why
In every single image we receive from the camera all the boats inside need to be detected.



These detections can be then further validated over time by using the existing calculation of the 3d position and tracking over time. As this second step can be used to validate the detections, we can output detections with a quite low boat-class confidence.

Details
The boats can be detected as simple rectangle boundingboxes. There are numerous approaches on object detection by using deep learning. You can check the following resources for state of the art algorithms:

- Kaggle challenge “Airbus Ship Detection Challenge” for some recent ideas on detecting boats in images
- Object detection in general: https://paperswithcode.com/sota/object-detection-on-coco

One-stage-detectors like YOLO, SSD seem to have problems to detect small objects like boats in our images. More recent one-stage-detectors like RetinaNet and two-stage-detectors like Faster-RCNN seem to perform better in this task.
One major tool to handle the small boundingboxes is using Focal Loss. You can find more insights and hints on the problem of detecting small objects in this blog post: https://mc.ai/small-objects-detection-problem/

Data
Bodensee
Airbus
DOTA

Evaluation
Average Precision with 50% IoU

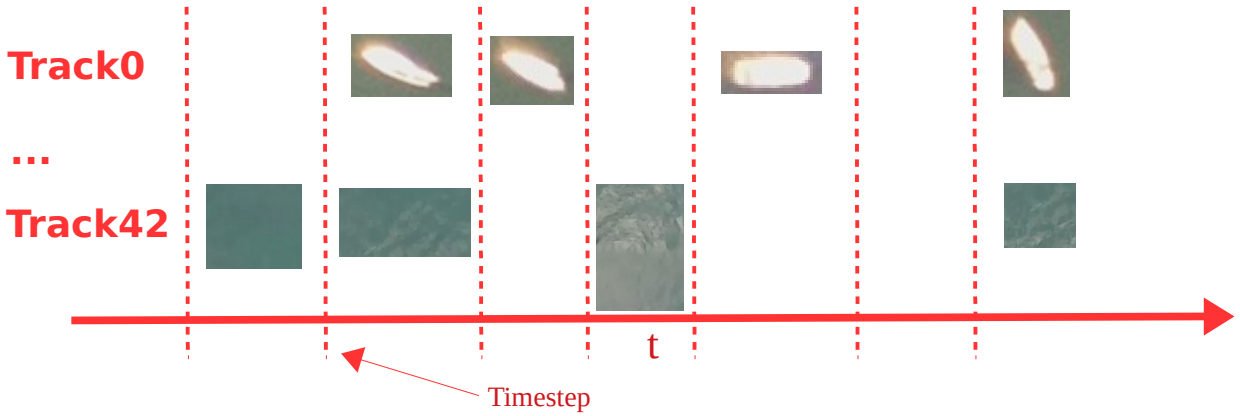
TEAM C

Multi frame – objectproposals / tracking classification

Needed Skills: Python, Deep Learning

Why

The existing code for the boatdetector can already detect proposals / regions of interest (ROIs) in every single camera image and associate them to each other to form trackings of them. The idea is now to classify not just single images, but multiple images from a track to a single track classification result. Therefore temporal information can be used for the classification of a track.



At every timestep, the trackings can be classified by using all the available images so far.
By using this temporal information we can get more stable classification results and lower the system load by deleting non-boat tracks more early.
The algorithms can learn two things this way: 1. try to actually differenciate between/classify boats and nature and 2. try to learn the similarity between the frames, as the nature images are constantly changing their appearance.
By classifying multiple images of a track, the overall accuracy of the detector should be slightly better than classifying single proposal images, as more information about a possible boat can be considered for the classification.

Details
The main challenge in this workpackage is to find a way to compose the images of a track in a meaningful way to create a classification problem. Some ideas to do this:

- Classify single images and combine the resulting class propabilities by a custom formula
- Use a recurrent network to take the temporal information of multiple images into account at the classification step: http://bmvc2018.org/contents/papers/0309.pdf

For the image classifier itself and the problem of different image input sizes, check the “Single frame – image and objectproposal classification” task for suggestions.
To calculate the similiarity between the frames, classical simple shape descriptors like “Hu Moments” could work as they are translation, rotational and scale invariant.

Data
Bodensee

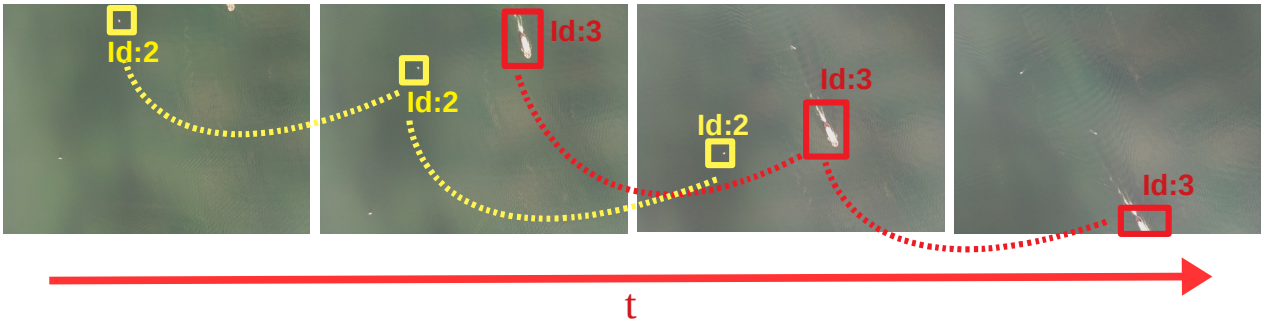
Evaluation
Average Precision

TEAM D

Multi frame – object detection with Tracking

Needed Skills: Python, Deep Learning, Tracking

Why



Beside the detection of boats on single images, it is also important to track them over time, thus to redetect them in multiple images. This is important, as we want to gather more information about single boats for a single transmission to the rescue ship. The tracking can be also used to reduce false positive detections, as boats do not change their shape in contrast to waves.

Details

The detector can be implemented like the one from the “Single frame – object detection” workpackage.
One approach to create a better single frame detector is using recurrent networks to fuse feature information across multiple frames:
http://bmvc2018.org/contents/papers/0309.pdf

Beside using non deep-learning based trackers, a tracker could be implemented by using Siamese-Networks like in here:
https://arxiv.org/pdf/1901.01660.pdf

Another approach would be to create a single deep learning network, which is capable of doing both – detection & tracking:
https://arxiv.org/abs/1710.03958

Data

Bodensee

Evaluation

Detection
Average Precision with 50% IoU
Tracking
Repeating Track-ID Accuracy

TEAM E

Object detection via anomaly detection

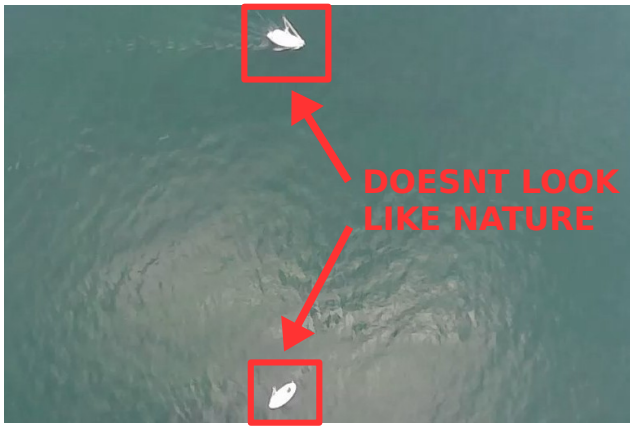
Needed Skills: Python, Machine Learning, Deep Learning

Why

There is the problem in the project that we have very little actual data with real refugee boats on the sea, which we can use to train a object detection algorithm. This problem can possibly avoided by tackling the detection from a different side. Instead of trying to detect actual boats, we can also try to detect things which for sure do not look like nature, by training a algorithm how nature at sea looks like. This way we do not necessarily need lots of data from actual boats. There are two different anomaly detection approaches possible:

Anomaly proposal classification

Anomaly object detection



Details

There are different implementations of this algorithm possible. One keyword to search for is "One-Class Classification". SVM-like algorithms seems to be popular in this area. Its also possible to use deep learning for this task.
You can also check out One-Shot-Learning using siamese networks, which can help on the problem of having little data of actual refugee boats.

Methods not involving machine learning which look promising for this task involve traditional pattern recognition:

- "The Phase Only Transform for unsupervised surface defect detection", Dror Aiger, 2010 – Fast due to FFT!
- "Reducing Anomaly Detection in Images to Detection in Noise", Axel Davy, 2018 – Might be too slow!

Data

Anomaly proposal classification + Anomaly object detection
Bodensee
Airbus
DOTA

Evaluation

Anomaly proposal classification
Average Precision
Anomaly object detection
Average Precision with 50% IoU

TEAM F

Converter for real recorded drone mission data to ROS-format

Needed Skills: Python, drone software

Why

The current software on the drone is running a own implementation for the image grabbing to the companion computer. This implementation saves GPS data like position and time to the EXIF area of the image. This data is received through self programmed MAVLINK interface.
Additionally to this data, we need the exact position in meters in regard to the start point of the drone and the orientation of the drone, to estimate the exact position of found ships. This data is saved in the PX4 log files of the drone.
As we got some mission data already which is recorded in this format, it would be good find a way to use it in the ROS environment we are currently using to run our image recognition task. Thus a converter is needed to take the images with the timestamps and the log files with additional drone position / orientation data and transform this data to the ROS format.

Details

To parse data from the PX4 logfiles, there are already some libraries available, like pyulog. Data from the EXIF area of the images can be read by using libraries like piexif.
The format of the data in ROS needs to be as following:

Data	ROS message type
Local position / orientation	geometry_msgs/Pose
Global position (GPS)	sensor_msgs/NavSatFix
Image	sensor_msgs/CompressedImage

As the drone internally uses different coordinate system conventions to handle the local position / orientation data than ROS, the data needs to be transformed to be usable in ROS. You can read more about that over here:
https://github.com/mavlink/mavros/blob/master/mavros/README.md#coordinate-frames
MAVROS is a software which takes care of all conversion of MAVLINK data to ROS. But it can only be used for live MAVLINK data.

Data

PX4 / image mission data
https://www.hs-augsburg.de/homes/beckmanf/searchwing/flug20190519/picam/

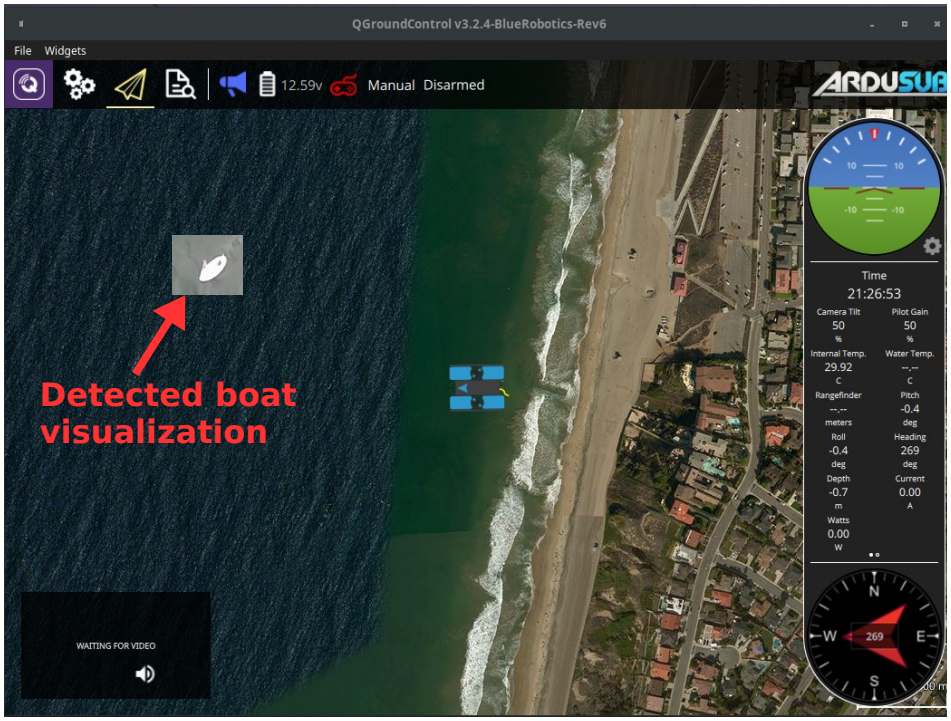
TEAM F

MAVLINK image sender to rescue vessel

Needed Skills: Python, drone software, (C++)

Why

When running the boatdetector on the drone, we need a way to send and visualize detected boat images and its metadata on the rescue vessel. As the communication between the drone operator and the drone is only possible via the telemetry link, we need to send all the detected boatimages via this link. The protocol used on the telemetry interface is MAVLINK. All the received images need to be visualized in a user friendly way.
The visualization could be either hand made, showing all the detected boats on a map, or integrated into existing drone monitoring software like QGroundcontrol.



Details

MAVLINK supports sending of images: https://mavlink.io/en/services/image_transmission.html
As we also need to send metadata like GPS position and time for the detected boats, we need to send the image including the metadata in the EXIF-block of a jpeg image.
To simulate detection images we can use pymavlink to create image messages and send them over a TCP connection to a receiver: https://github.com/ArduPilot/pymavlink/blob/master/examples/mavtcpsniff.py

The visualization of the received data need to include the image (on a map), and a possibility to see the metadata. QGroundcontrol (C++) supports receiving of images over a TCP interface, but not over MAVLINK. This code would need to be implemented. Some information how to implement this can be found here: https://mavlink.io/en/services/image_transmission.html#developer

TEAM G

Evaluate Deep Learning on embedded Platforms

Needed Skills: Embedded Hardware, Deep Learning

Why
In the end, the developed algorithms need to run on the drone on a very energy efficient embedded system. The hardware needs to be able to run the detection algorithms on the fullsize camera image in acceptable framerates at about 1 FPS. Currently there are several embedded systems on the market, dedicated to run deep learning algorithms in a efficient way. These should be tested.
Beside the hardware, there are also different frameworks to run deep learning algorithms, which need to be evaluated.
To evaluate the platforms, the team should talk to other teams for meaningful neural networks to evaluate at the specific platform. The power usage should be estimated too, to estimate the efficiency of the whole platform.

Details
There are different platforms available at the hackathon for evaluation:

- Pi4 / Pi3
- Jetson Nano
- AIY Vision Kit (Movidius 2450 + Pi Zero)

Possible deep learning Frameworks to evaluate:

- Tensorflow
- Tensorflow Lite
- Tensorflow TensorRT
- Arm Compute Library
- OpenVINO
- TVM Compiler
- Tencent NCNN

Parameters to evaluate:

- Framerate
- Powerconsumption

To estimate the powerconsumption, I provide a power meter.

Data
Use data which fits to the network you want to evaluate