Computer Vision

In order to identify the aircraft the ROV need to be able to distinguish between the different models. The problem is tackled using a color detection algorithm. The flowchart of the algorithm can be viewed in figure (). The algorithm uses functions from the open source library OpenCV.

# Flowchart explanation

## Camera

The camera module are controlled by a nano-pi. This nano-pi creates a g-streamer pipeline using the command below.

“gst-launch-1.0 v4l2src device=<DEVICE> ! video/x-h264, width=<WIDTH>, height=<HEIGHT>, framerate=<RATE>/1 ! h264parse ! rtph264pay config-interval=-1 ! udpsink host=<TOPSIDE\_IP> port=<PORT>”

The tags are filled in with appropriate values. This pipeline is streaming the camera feed to a port <PORT> at the specified <TOPSIDE\_IP> ip address over ethernet.

## G-streamer pipeline

At the receiving side we initiate a new GStreamer pipeline streaming the camera feed on the specified <PORT> to the plane detection algorithm. This pipeline is initiated by the VideoCapture class in the OpenCV library.

## Reading the frame

In main.cpp line 13 the pipeline is specified to the algorithm. Then the algorithm initialize a while loop which start by calling the “activate” function from vision.cpp. This function reads an image from the pipeline and gives it to the object m\_frame. The next step in the main.cpp step calls the “analyse” function from vision.cpp which does the visual processing on the image.

## Color detection

The color detection part of the algorithm has a BGR-image as input and output three mask images. The mask images are binary images which has the pixel value of 1 or 0 depending on whether the pixel value of the input value are within the specified bounds.

The algorithm operates in the HSV, Hue Saturation Values, color representation. In order to convert the input image to hsv values the function cv::cvtColor() is used on a blurred version of the input image. This new image is then subjected to the cv::inRange() function which takes the specified color bounds and creates the mask image. There are three mask images that are created, one blue, one red and one yellow.

## Shape detection

The shape detection part of the algorithm take a mask image as input and output two vectors containing the information of any detected squares or triangles. The algorithm runs the shape detection on all three of the mask images separately.

The algorithm start by running the cv::findContours() function which outputs a vector containing the information of any detected contours. This vector is then searched through and if the area of the contour is outside the specified bounds the contour is discarded. This makes it so that small contours does not get taken into account.

The vector containing the approved contours are then searched again and evaluated based on its circularity which is calculated from the area and perimeter found with cv::arcLength and cv::contourArea. The contour is then sorted into the vector containing the information of any squares, the vector containing the information of any triangles or discarded. This evaluation is based on specified circularity minimum and maximum values for squares and triangles.

## Display bounding box and text

The display part of algorithm display a bounding box and text based on the information in the output from the shape detection part. The bounding box is the laid on top of m\_frame using cv::rectangle(). The text is also displayed on m\_frame using cv::putText().

## Screen

The last part of the “analyse” function in vision.cpp display m\_frame in the window “Analysed” by using cv::imshow().