# Assignment 2 – Intelligent robotics

Our task was to create interaction loop between iCub robot and visual input using YARP and iCub simulator. The program has a capability to detect face and circle, follow it with his eyes and is capable of making a simple gesture when he recognises some specific object.

Yarp (Yet Another Robot Platform, http://www.yarp.it/) is responsible for a communication with robot in a peer-to-peer way using most popular connection types and using Observer design pattern. We are extensively working with this platform to obtain robots visual perception or transfer USB camera feed to robot.

All computer vision tasks were handled using OpenCV library (we tried to use ArUco library for augment reality application but there we hit a problem with linking libraries). OpenCV is a computer vision library (<https://opencv.org/>) that allows for an efficient real-time application. We use it for face detection with Cascade Classifier and circle detection using Hough Circle Transform.

# Step 1 and 2

Thomas work here

# Step 3

We have implemented 4 different linear filters.

First one, from method Detector::applyLinearFilter(ImageOf<PixelRgb>\* yarp\_img) is just an example from OpenCV documentation that is just blurring an image using all-one matrix.

Second filter Detector::edgeDetectionFilter uses kernel from <https://en.wikipedia.org/wiki/Kernel_(image_processing)> that is supposed to detect edges, however in current implementation it mostly just darkens an image.

Third one, Detector::cannyEdgefilter uses Canny edge detection, know as well as the optimal detector.

Fourth one, Detector::sobelFilter uses Sobel Derivatives that using approximations of derivatives and other hair-raising math, yields similar results to Canny, however is not that accurate.

Unfortunately, as we encountered some problems with port conflicts, we didn’t manage to display image to another yarp view window, instead we save each image to .jpg file with descriptive name.

# Step 4

After filtering image from camera using one of the functions from step 3, we were capable of using

Haar Cascades for face detection. Haar Cascades are cascades of classifiers that are looking for a human’s face features, as they are pre-generated using machine learning algorithms. In our example implementation we are calculating a centre of face and then drawing an ellipse around face to be sure that algorithms actually works.

This function is also an entry point for step 6.

# Step 5

The goal of step 5 was to implement marker detection using ArUco library or simple circle detection. Unfortunately, we were not able to link ArUco libraries using CMake (I believe there must have been some problem with their compilation) so we decided to implement circle detection.

Circle detection has been implement using Hough Circle Transform and OpenCV method called HoughCircles. To be sure that the function actually works the output image has drawing on it with red circle and green centre of this circle that overlaps detected figure.