Computer Vision - Assignment 2 - Report

Paul Asquin, CentraleSupélec MSc in AI March 2019

Introduction

This project aim was to develop a method to identify humans moving in front of a ground-fixed camera without using Deep Learning methods. In order to make such think possible, we had to work extensively with “classical techniques” in computer vision, and take advantage of the OpenCV library.

We will consider our project having, as input, the list of images where want to find humans, and as output, an list of shape (number_of_frame, id_of_box_in_the_frame, box boundaries), that will describe where are humans in the succession of images.

Data pipeline

The most important point of this report is to present the data pipeline and how it ends up achieving the person recognition: here is the detailed process.

Background substraction

Background substraction from image 287

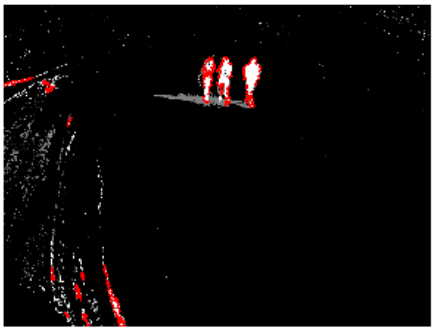
The OpenCV background substractor definitively is a great tool for this project: we created a *createBackgroundSubtractorMOG2* object and applied images to it, allowing the object to generate masked-images such as the one displayed on the top. The background substractor puts in evidence the moving entities: indeed, we can see the

pedestrians moving, but we have a lot of noise, from the tarp moving with the wind to the people shadows. We will have to combine this tool with others.

Get contours

In order to work with the background-substracted images, we have chosen to work with contours as a start for the box finding. After having resized and then applied a *GaussianBlur* to the images in order to rub out the resize effects, we used the *findContours*. By drawing every contours in the same image, we can obtain results such as this one:

Contours finding from image 287



Indeed, we managed to extract the humans, but we also extracted part of the tarp. We will create a first step of boxes with the results of this methods: 1 box correspond to the boundaries of 1 contour.

Box merging

Sometimes, the same person can be cut in multiple contours: legs, bust, head… All split apart. In order to override that kind of problem, we have chosen to merge together two boxes that would be close one of the other, and stack one on the top of the other. This way, we are able to have 1 unique box per human, and still boxes with part of the tarp.

Box ratio filter

Humans are not cubic: unless if pedestrians are crawling on the ground, they should be taller than larger. This way, we eject from or list of potential-human-boxes the ones that haven't a height/width ratio between 1 and 12. This lead to a score of ~0.18, that we want to improve with other techniques.

HOG & SVM

We wanted to have a SVM guessing if what its seeing is a human or a random background sample. Then, we have implemented dataset generation function

Positive (human) and negative (background) dataset

We generated datasets made of 15px*30px images containing pedestrians in the first case, or random background element (as long as they don't overfit on humans) to train a binary SVM classifier.

SVM Training

The SVM training has been done by appling first a *HOG* transform to the images as a way the reduce the information in the input pipeline. This led also to flatten data, required for or SVM. At the end of the day, our SVM managed to reach awesome pedestrians recognition skills with an accuracy of 0.93 on the test set.

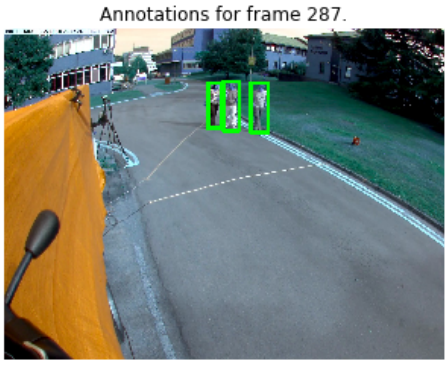
Inertia consistency

We have implemented an *inertia checking* function as a tool to check if a box have a continuity in the space and time: that means a pedestrians couldn't be teleport from one part to the other of the image, then every box is likely to have another box near to it after some frames, probably the box describing the same pedestrians. This could have been great with pedestrians, bu turned out to be inefficient, as tarp vibration was also continuous. Then we have chosen to deactivate this pipeline.

Outputs, pros, cons and possible improvements

We managed to get a score of 0.42, which is quite great considering a non-deep-learning implementation. We was indeed quit surprised by the capacity and relative fastness of OpenCV implementation. We generate a video output in order to analyze the behaviour of our pipeline: the pedestrians aren't well recognize when they are quite at a long distance from the camera, as the image quality is very low. Still, when they are getting closer, we definitively are able to well defined the box that are fitting to them, and this went even better thanks to the SVM classifier. In some rare case, we can still see the tarp-area activated, as part of the error of the SVM classifier.

Proposed solution for image 287



Still, if I could have put more time on this project, I would have highly wanted to train the SVM on a lot of pedestrians images. Indeed, I haven't managed to take the time to extract pedestrians from the Caltech dataset. I am using pedestrians from the video to recognize pedestrians from the same video: this is a really big problem, as I could have highly overfited on the characteristics of the 3 persons walking. I hope the SVM classifier still is capable to identify other humans, and that it will perform great on your test dataset. I haden't the time to do it, but I totally know that I should have fed my SVM with other positive and negative samples than the ones from the provided dataset.