Immagine che contiene edificio, esterni, facciata, pietra

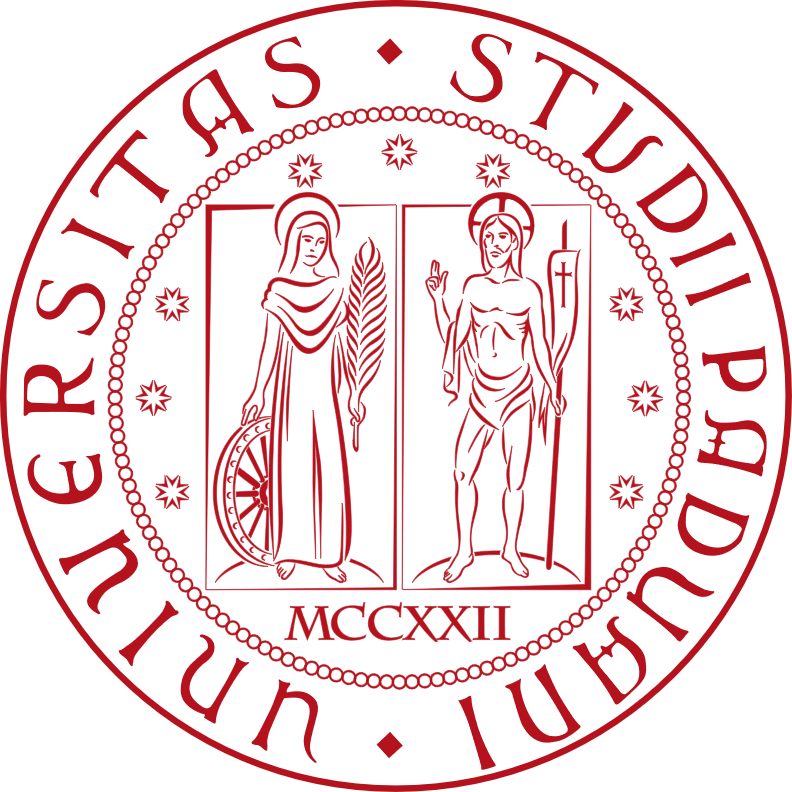
Descrizione generata automaticamenterettangolo bianco per il testo sul frontespizio

Summer project 2022

Hand detection and segmentation

Computer Vision course

A.A. 2021/22

Prof. Stefano Ghidoni

**//TODO number of working hours per person**

**Potremmo mettere una media delle ore lavorative all’ultima pagina**

**// Your report must include the results of the hand detection and hand segmentation (both output images and metrics values) for the test images in the provided benchmark dataset (according to the number of members of your group) ….???**

**Cioè? Ci schiaffiamo tutti gli output della detection e della segmentation così come immagini sul report? Quindi 60 immagini??**

**Oppure mettiamo direttamente il link del drive alle cartelle???**

**//TODO se riusciamo a caricare file su moodle togliere link**

**Link to Google Drive directory with source code, report, and results:**

<https://drive.google.com/drive/folders/1oCrcbEfxttbjuv5dxdHW9ACO2LxGUmWI?usp=sharing>

**Introduction**

The goal of this project is to develop a system capable of

1. detecting human hands in an input image
2. segmenting human hands in the image from the background

The solution proposed is based on training a Neural Network for point 1 and using the OpenCV library for point 2.

**Developing a hand detection program**

Since the idea is to train a Neural Network (NN) to detect if an image contains hands and where they are, the first thing that we need is a dataset.

The test dataset consists of 30 images:

* the first 20 images are from the EgoHands dataset.[[1]](#footnote-0)
* the last 10 images are from the HandOverFace dataset.[[2]](#footnote-1)

The training dataset will be discussed later.

The approach that we want to use to train the NN is called Transfer Learning, which consists in using a pre-trained model that was previously trained on a large dataset. The intuition behind transfer learning is that if a model is trained on a large and general enough dataset, this model will effectively serve as a generic model of the visual world. You can then take advantage of these learned feature maps without having to start from scratch by training a large model on a large dataset.

For performing transfer learning, we need a custom dataset, in our case hands‘ images.

We end up using the YOLOv3 network, which is a multi-object detection algorithm that uses a convolutional neural network (CNN) to detect and identify objects.

**Creating a training dataset**

Since we want to fine-tune the YOLOv3 network, we need a training dataset composed of hands images. Since the test dataset contains images from two different datasets, we built a custom training dataset in the following way:

* **EgoHands dataset**
  1. Download all the labeled data, which consists of 4800 images, and, with the Matlab script provided by the EgoHands dataset authors[[3]](#footnote-2), generate the ground truth for each image (i.e., a *.txt* file which contains the pixel coordinates of the bounding boxes in the COCO format[[4]](#footnote-3)).
  2. Check if there are duplicates between the train and the test datasets and delete them.
  3. Convert all the *.txt* files containing the ground truth from the COCO format to the YOLO format[[5]](#footnote-4).
* **HandOverFace dataset**
  1. Download a dataset that contains images like the ones in the test set[[6]](#footnote-5).
  2. Check if there are duplicates between the train and the test datasets and delete them.
  3. Since the ground truth of the dataset downloaded are segmentation masks, we need to generate the bounding boxes. For this step, we used a useful tool that allows us to manually select the area of the image where is the object, in our case the hands, and it generates the *.txt* file with the bounding boxes values[[7]](#footnote-6).

In the end, the training dataset is made from the two previous datasets combined. This dataset can be found at the following link: <https://drive.google.com/drive/folders/1zhCNISW0AfA412r51QMFAUho_QMav6HM?usp=sharing>

**Training a custom object detector**

Since we don’t have the powerful hardware needed to train a NN, we decided to use a powerful tool provided by Google, which is Google Colaboratory. This tool allows us to run a Jupyter Notebook in the cloud, using powerful GPUs.

For training our custom network we followed the documentation of the YOLO author[[8]](#footnote-7).

The code and its explanation are in the ***trainingALL.ipynb*** file in the ***NeuralNetwork*** directory. The other files in the directory are the files needed to train the NN.

Once the training is done, we need to save two files:

* ***yolov3\_custom.cfg***, a configuration file that tells us how the network is done.
* ***yolov3\_custom\_final.weights,*** which contains the weights of the network.

**Importing in C++ the Neural Network**

Using the DNN module provided by OpenCV[[9]](#footnote-8) we can import a trained neural network.

Following the tutorial provided by OpenCV[[10]](#footnote-9) we developed the ***Detection.cpp*** program that does the following steps:

1. Loads
   * the configuration file ***yolov3\_custom.cfg***
   * the weights file ***yolov3\_custom\_final.weights***
   * the ***classes.names*** file which contains the name of the classes, in our case only *hand.*
2. Load the NN using the readNetFromDarknet() OpenCV function.
3. Propagate the weights with the forward() function and save the output in a cv::Mat variable.
4. Processing the output to get the confidence, the class detected, and the values of the bounding boxes, which will be saved.
5. Draw the bounding boxes on the input images and save them.

**Segmentation by Tommaso Martone**

The target of the segmentation was to detect each hand in a dataset of thirty images. It is a challenging task due to several problems that have been encountered during the evolution of the project. In general hands in a picture will appear in different positions, sometimes partially covered by objects, presented from different points of view, and also can appear in different sizes. It also has to be considered that skin color is unique for each person and even light conditions can make the hand lighter or darker.

For these reasons, at the beginning has been tested different segmentation algorithms in order to evaluate which one can give a good raw result and then improved with some adjustments.

At first, it has been tested K-means algorithm, but the optimal amount of clusters needed to properly segment the hands drastically change from one image to another, for this reason, it has been selected another algorithm that avoids any knowledge a priori of the number of clusters: mean shift. Mean shift after some pre-filtering provides better results compared to K-means but still requires more work.

The hands in the pictures cover a small area of the image so sometimes the hand’s clusters were merged into similar or close clusters making it difficult to proceed with the segmentation. After that, it has been used the window obtained from the previously performed detection of the hand and applied the mean shift algorithm to the restricted area. In this case, since the cropped image is much smaller even a little variation of light on the hand can lead to segmenting the hand into different parts. So, it has been performed the mean shift before the crop phase on the whole image.

Subsequently, it has been considered that now the hand covers almost the entire image window, and then it remains to isolate the background from the hand. To solve this problem it has been exploited the Otsu thresholding method, adapting the algorithm and introducing a second threshold in order to obtain the hand segmentation between the two thresholds. But also in this case shadows and lights drastically affect the image even by introducing a smoothing filter or different kinds of filters before the application of Otsu's method.

To solve the light problem researches has been done on how to face this problem and make the image as much invariant to light conditions as possible. Evaluating the HSV color space, and more specifically, tuning the brightness channel, it has been take into consideration the paper: “Human Skin Detection Using RGB, HSV and YCbCr Color Models” by: S. Kolkur , D. Kalbande, P. Shimpi, C. Bapat, and J. Jatakia. In this study, the hand segmentation problem is tackled by evaluating the same image under different color spaces and then selecting a range of values for each channel of each color space, so it has been decided to test this algorithm introducing some adjustments.

In a first attempt encouraging results were obtained, but still, light plays an essential role in the segmentation. Normalizing the V channel of HSV space improved the segmentation but for many images the background is too bright or too dark with respect to the hands making the segmentation impossible in some cases. For this reason, it has been performed the equalization of the brightness exploiting the window obtained by the detection in order to have an optimal exposition of the hands. To find the best value for the equalization it has been introduced a CallBackFunction on the images that at each event print out the values of the V channel.

The next step was to perform the thresholding on the skin tones. In the previously mentioned paper, it has been evaluated two different solutions, one using RGB and HSV spaces and the other using RGB and YCrCb spaces, the first option has been selected. The whole implementation of the algorithm of the paper does not returns the expected results but manually tuning the parameter of the thresholds has been reached sufficiently good results.

The conditions imposed by the thresholds lead to detect also pixels with colors similar to the skin tone, for instance, chessboards or some small object. To restrict the region of interest (RoI) it has been used the detection window used before for the equalization of the V channel, but this time keeping unchanged the RoI and painting in black all the pixels outside them.

Introducing a concatenation of erosion and dilation functions for smoothing the edges and removing impurities, the resulting image can be used as a mask for hand segmentation.

Even if the algorithm does not exploit any complex computation it can return a quite high accuracy on the hand segmentation with the advantage of being light and fast.

The whole algorithm can be reduced to the following steps:

* Equalize light brightness with respect to the hands.
* Threshold the image on RGB and HSV color spaces.
* Create the mask.

**Segmentation by Matteo Rambaldi**

The initial approach to the segmentation was to define a set of image preprocessing for reducing image noise and blurring it. After that it has applied and tested different types of segmentation algorithms, to search for better results on the different kinds of images in the test set.

The first attempt was to apply an initial Mean shift blur, image edge preserving filtering algorithm that is frequently used to eliminate noise before image watershed segmentation, which can considerably improve the watershed segmentation effect. A multi otsu’s thresholding to define two different thresholding values, applied to segment the region of the image with intensity more equal to the skin. To the binary mask has been applied a morphological transformation as erode and generate the distance transform image. From that has been used the watershed algorithm, but this kind of segmentation led to an overflow of the accuracy to the first images and to a decrease in the remaining dataset.

The second attempt has been to apply the k-means algorithm, with similar preprocessing to the image as before (blur smoothing, histogram equalization). This approach is used to group similar colors into defined clusters. But for thresholding the hands in the image to get a final binary image of the objects of interest, it was too difficult to find the perfect match of number of clusters and iterations for all images, also for the big variety of colours in the scene.

The third attempt has been to apply Simple Linear Iterative Clustering (SLIC), an efficient superpixel segmentation method for extracting superpixels with greater adherence to the edges of the regions, composed by linear iterative clustering and graph theory-based algorithms. But the algorithm was too complicated to understand the code of the base idea.

The ultimate attempt has been to prove with a different approach defined in the paper: “*Identification of Hand Region Based on YCgCr Color Representation by YaNan Xu and Gouchol Pok*”. This approach processes the input image initially to balance the light source of the image, adjusting the value of B, G, R with the illumination intensity of the scene. Then work in the YCgCr color space, so that brightness and chrominance are separated and with a fixed thresholding is it possible to determine the skin area. At this point the complex of the background produces noise and skin-line, so the idea is to remove the non-gesture regions. The problem of this application was the difference in the skin colours for different etnies in the thirty images of the test data, that led to a not good segmentation with the fixed thresholding and also the problem with one image of the set that is originally provided in the grayscale.

**Performance measurement**

* Intersection over Union (IoU) on hand detection:

for measuring this kind of metrics has been needed to use the coordinates of the ground truth bounding boxes and of the our predicted bounding boxes. The base idea for calculating the IoU has been to calculate the overlapping area between the two bounding boxes and divide it to the union area of them. The coordinates of the predicted bounding boxes were saved in different order despite the ground truth, so to match the bounding box was necessary to compare the more similar coordinates, and calculate on it the metric.

* Pixel accuracy on both classes (hand, non-hand) for hand segmentation:

//TODO

**Usage of the program**

The program will compile using CMake and pass as argument one of these two ways:

* A path to a file ***.jpg***, which will display the detected and segmented image to the user. The bounding boxes and the segmentation masks will be saved.
* A path to a directory containing one or more ***.jpg*** files, which will save all the detected and segmented images with all the bounding boxes and the segmentation masks.

For a more soft use of the program has been introduce a simplest menu for choosed the action:

* 1 -> to obtain the detection.
* 2 -> to obtain the segmentation.
* 0 -> for exit.

**Work done by each member**

Each member actively participated in the development of the project contributing with ideas and discussing the results obtained in all phases. It is possible to divide the workload done by each one as follow:

* ***Tommaso Bazzan***, ID: 2062447
  + Generating the two training datasets.
  + Developing and training the custom hand detector ***trainingALL.ipynb***.
  + Developing the ***Detection.h*** and ***Detection.cpp*** files.
* ***Matteo Rambaldi***, ID: 2055531
  + Different approaches to segmentation.
  + Developing the ***Segmentation.h*** file.
  + Metrics of detection.
* ***Tommaso Martone***, ID: 2016972
  + Segmentation based on color skin and other approaches.
  + Developing the ***Segmentation.cpp*** file.
  + Metrics of segmentation.

1. <http://vision.soic.indiana.edu/projects/egohands/> [↑](#footnote-ref-0)
2. <https://drive.google.com/file/d/1hHUvINGICvOGcaDgA5zMbzAIUv7ewDd3/edit> [↑](#footnote-ref-1)
3. <http://vision.soic.indiana.edu/egohands_files/egohands_data.zip> [↑](#footnote-ref-2)
4. The COCO format is: [x\_min y\_min width height] [↑](#footnote-ref-3)
5. The YOLO format is: [class x\_center y\_center width height] [↑](#footnote-ref-4)
6. <https://www.kaggle.com/code/rustyelectron/prepare-handsoverface-hof-dataset/data> [↑](#footnote-ref-5)
7. <https://github.com/heartexlabs/labelImg> [↑](#footnote-ref-6)
8. <https://pjreddie.com/darknet/yolo/> [↑](#footnote-ref-7)
9. <https://docs.opencv.org/4.x/d6/d0f/group__dnn.html> [↑](#footnote-ref-8)
10. <https://docs.opencv.org/3.4/d4/db9/samples_2dnn_2object_detection_8cpp-example.html> [↑](#footnote-ref-9)