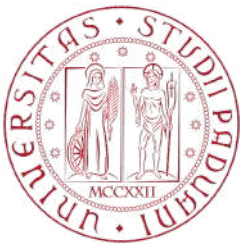
Computer Vision

Project report

Hand detection and instance segmentation

Francesco Gazzola



2022

0.1 Methodology

0.1.1 Detection

The detection module has been achieved by focusing on the shape of the object to detect. For extracting such information, we computed the HOG features for horizontal and vertical hands and trained two SVMs over the HOG features in order to detect horizontal and vertical hands respectively. In this way we obtained two HOG-based detectors.

The detection is then performed using the sliding window technique with a window stride of 16 pixels along both x and y direction, over a pyramid scale space in order to detect the hands at different scales. The window size of our final solution is (40x56) for the vertical hands detector and (112x72) for the horizontal hands detector. The predicted boxes are then processed with a no-maxima suppression.
The two lists of horizontal and vertical hands are then combined together and processed again with a no-maxima suppression. The resulting predicted boxes are in turn classified with a skin detector and thus those that are not containing any skin pixels are discarded. Eventually, we applied an inverse threshold based on the area of the boxes i.e we keep only the boxes whose area is less than 25000 $pixels^2$. The threshold has been found empirically.

This pipeline let us reduce the number of false positive.

Dataset set up

For training the SVMs we needed some positive and negative samples.

The positive samples have been created by considering the images from the dataset 'hand_over_face' and also the 70% of the images of the egohands dataset. In particular, we used the egohands dataset provided by Roboflow.com [1] which comes already split in train, val and test set with xml files as annotations. From both the dataset we removed the images we used for testing our solution. Then, we resized all the images so that all of them

had width=600 and height computed so that they preserved their original aspect ratio. In performing the resize operation we also updated the annotations of the boxes.
Thus, for building the positive samples we randomly cropped out the regions of the images containing hands, which were identified by the ground boxes. We hence cropped along the size of the ground boxes with the adding of some padding in order to improve the ability of our detector to detect hands along the border. We then performed some preprocessing: convertion to grayscale, histogram equalization, resizing to the window size and bilateral filtering for preserving the shape of the hands and denoise.
We also used augmentation for the vertical positive samples (flipping around x, y and both axes).

Regarding the negative samples, we have built it by extracting from the images, some regions of the size as the size of the window size of our detector to train. We then filtered out these regions by keeping only those that had a zero IoU-score computed with respect to all the ground boxes of the corresponding image; and applied some preprocessing operations: convertion to grayscale, histogram equalization, bilateral filtering. Building the negative samples in this way, it gave us the possibility to extract information about the scene where the hands would appear.
The positive and negative sample step has been carried out two times: one time for considering only the horizontal boxes and one time for considering only the vertical boxes.

After gathering the samples, we computed their HOG features and trained two SVR (Support Vector Regression) with linear kernel, parameter C = 100 and epsilon = 0.1 .
We report in the table below the number of positive and negative samples we built for our final solution.

	VERTICAL SAMPLES	HORIZONTAL SAMPLES
POSITIVE	6128	7008
NEGATIVE	6293	7236

HOG features set-up

The HOG features are described in opencv by 4 parameters: the block size, the cell size, the window stride, the window size and the number of bins used for computing the histogram of orientation. For our experiments we keep constant some parameters while we tried different sizes for the window with aspect ratio representing the aspect ratio of the object to detect. We set up the parameters as follows:

- Block size = 16x16
- Block stride = 4 pixels for both direction x and y
- Cell size = 8x8
- Number of bins = 9
- Windows sizes: for our experiments we used the following pairs of vertical and horizontal windows sizes:

  - Vertical: 80x104 40x56 40x56
  - Horizontal: 112x72 56x40 112x72

Skin detector

The skin detector has been implemented by applying the algorithm suggested in the paper [2] for finding the skin pixels. The method suggested by the authors consists in checking a pixel over different colour spaces: RGB, HSV and YCbCr. The check is done by comparing

the colour of the pixels with respect to some thresholds proposed in the paper.
The classification is the performed based on the number of skin pixels that are found.

This method resulted in reducing the number of false positive.

No object detection management

We managed the case in which no hands occur in the image by checking the number of predicted boxes returned by the detector. A list of zero predicted boxes means that no hands were present in the image.

0.1.2 Segmentation module

We compute the instance segmentation starting from the predicted bounding boxes. We experimented three segmenting techniques:

- Contour-based segmentation
- Grabcut method
- Segmentation by thresholding over colour space

The method that resulted to be more accurate has been the grabcut and thus we used it for our final solution.

Contour-based segmentation

The first approach we used has been based on finding the contours of the object detected. For each predicted box we thus performed the following steps for getting the mask:

1. Crop out the region of the image identified by the predicted box and convert it into grayscale
2. Compute the edges through Canny algorithm on the region computed at the previous step

3. Apply dilation and erosion operation over the computed edges

4. Find the contours and create a mask by filling in the region delimited by the contour with the intensity value 255

5. Dilate, erode and gaussian smooth the mask

6. Given the mask, we coloured in the original image the pixels whose corresponding pixels in the mask are set to 255

Grabcut method

The second approach we exploited is instead making use of the grabcut method for separating the foreground object from the background. We exploited the built-in opencv function for computing grabcut which returns a mask where the pixels are set to 0 if it is a background pixel; 2 if it is probable backgroun pixel; 1 if it is a foreground pixel and 3 if it is a probable foreground pixels. We thus convert this mask to a new mask by setting all the 2 and 0 pixels to 0 andall the 3 and 1 pixels to 255. Then we colour in the original image the pixels whose correspondingpixels in the mask represent a foreground object (intensity value = 255).

Segmentation by thresholding over colour space

We segmented the image based on the thresholds used previously for the skin detector and create a mask. We then perform dilation, erosion and gaussian smoothing over the mask as done in the work [3]. Hence, we used the mask for colouring in the original image the corresponding foreground pixels.

0.2 Metrics

We used the Intersection Over Union for testing the performance of our detector. For every predicted box, we associated to it the maximum IoU computed over all the ground boxes. For the segmentation, instead we used the pixel accuracy metric defined as :

$$\text{Pixel accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

where:

- True Positive (TP): number of pixel classified correctly as hands
- False Positive (FP): number of pixel classified incorrectly as hands
- True Negative (TN): number of pixel classified correctly as not hands
- False Negative (FN): number of pixel classified incorrectly as not hands

0.3 Results

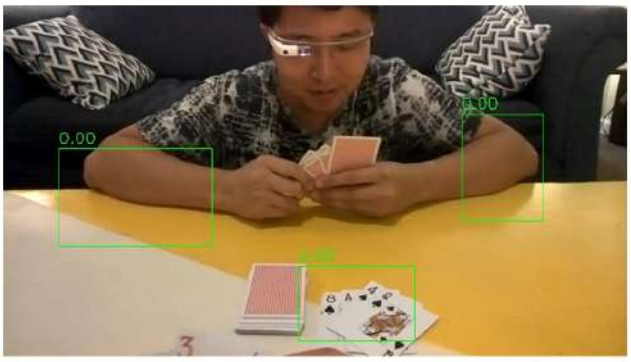
The first part of our worked has been based on finding a good classifier. After finding it, we moved to find the best segmentation techniques among those we proposed. Thus, we present here the prediction results we got by using different window size and the different outcomes obtained using the three segmentation methods.

0.3.1 Detection results

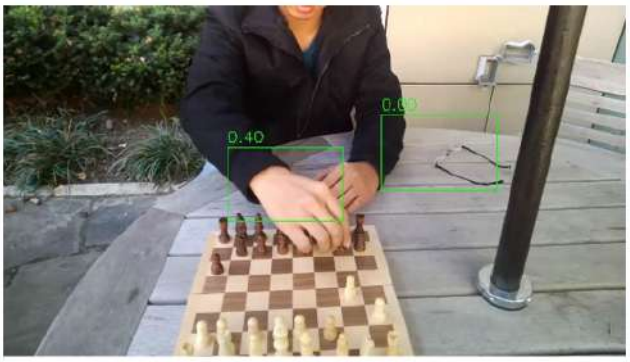
For each test image, we report its corresponding IoUs. The IoU scores are listed below each image in the same order with which the associated predicted box appears on the image from left to right. We printed the IoU scores also on the single box over the image for a better understanding.

Experiment 1

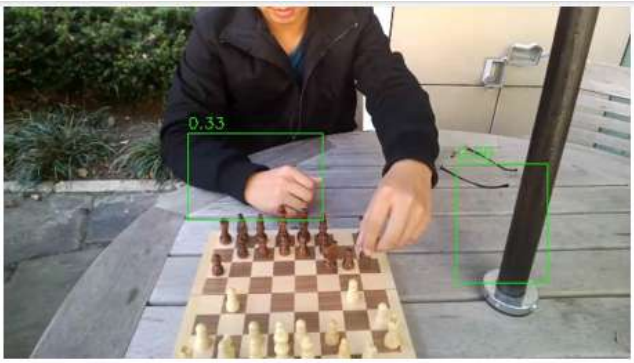
The first experiment has been carried out by choosing a vertical and horizontal window, whose size represented the average size of the vertical and horizontal ground boxes round to the closest integer multiple of the cell size. The rounding operation is required in order to compute the hog features correctly.
Hence the vertical window had size (80x104) and the horizontal window had size (112x72).



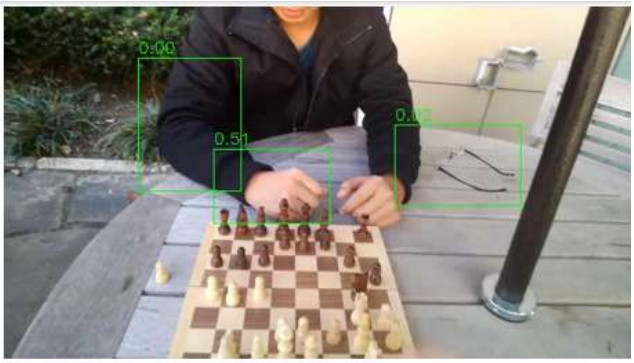
IoU = 0.00 – 0.00 – 0.00



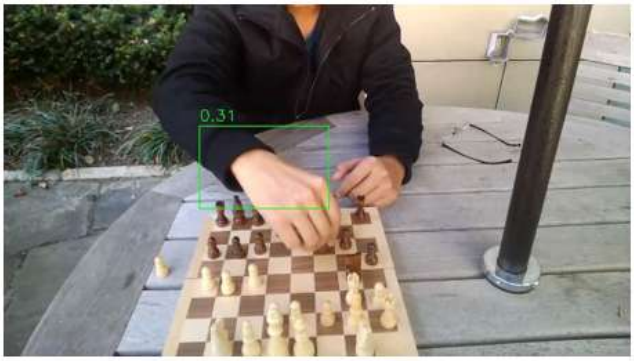
IoU = 0.40 – 0.00



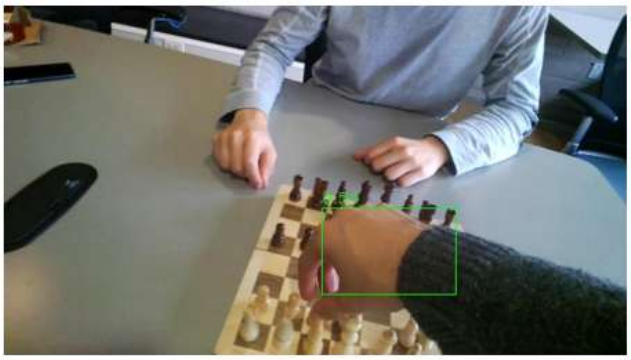
IoU = 0.33 – 0.00



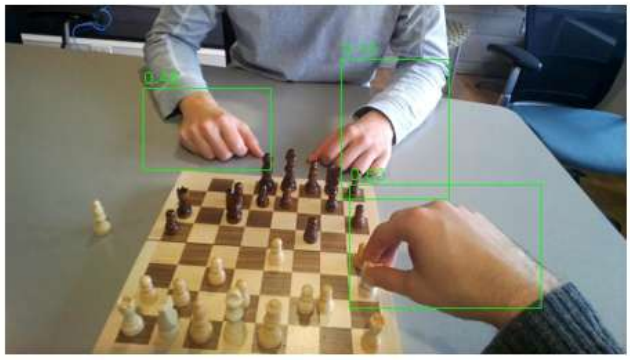
IoU = 0.00 - 0.51 – 0.02



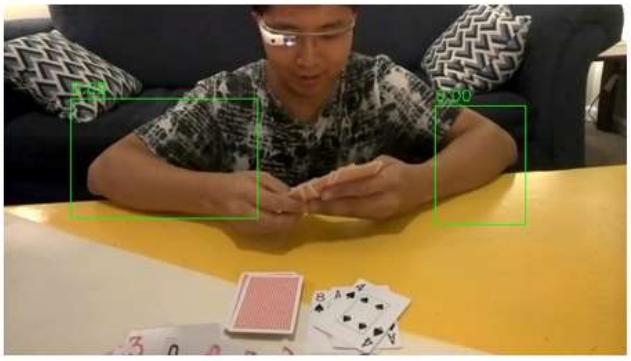
IoU = 0.31



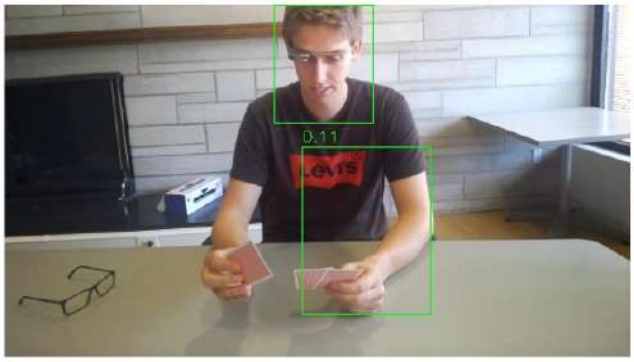
IoU = 0.54



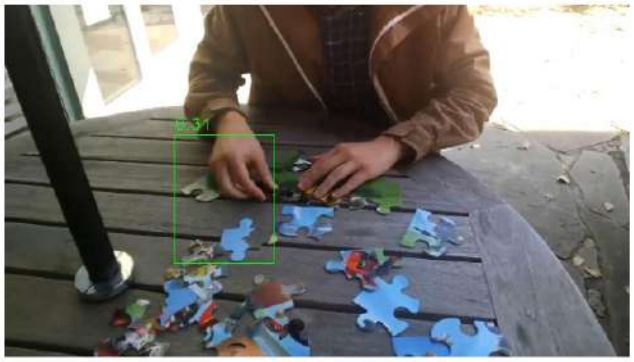
IoU = 0.48 – 0.65 – 0.16



IoU = 0.09 – 0.00



IoU =0.11 – 0.00

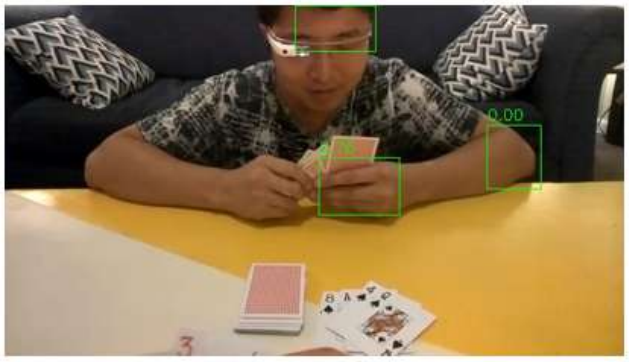


IoU = 0.31

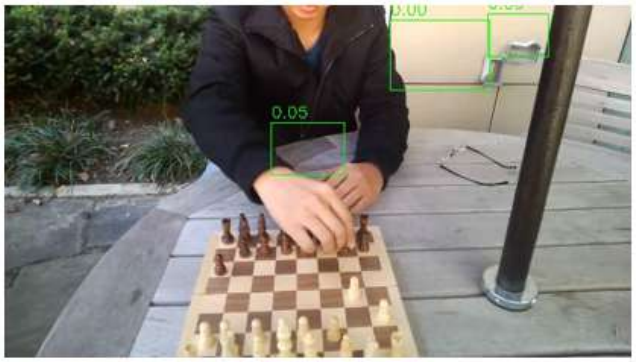
Experiment 2

Given the poor performance of the previous detector, we tried to decrease the size of the windows. The size of the windows adopted for this experiment as been determined by halving the average of the vertical and horizontal ground boxes and then consider the closest integer multiple of the cell size. Then, the vertical window had size (40x56) while the horizontal window size was (56x40).

Thus we recomputed the positive and negative samples and retrained the SVM.

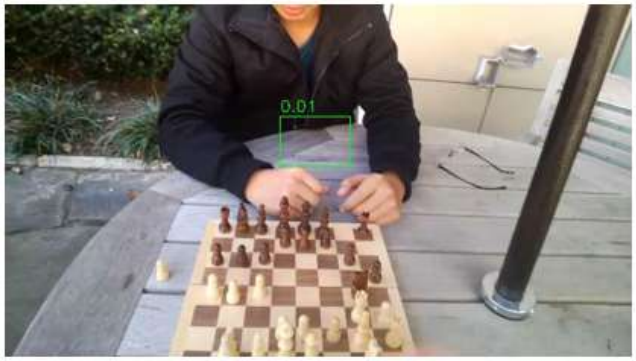


IoU = 0.00 – 0.76 – 0.00

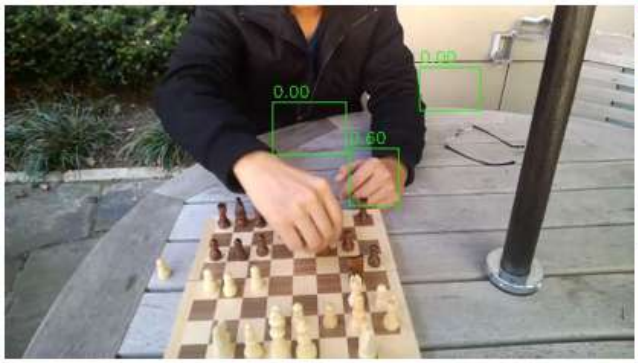


IoU = 0.05 – 0.00 – 0.00

IoU = 0.08 – 0.00



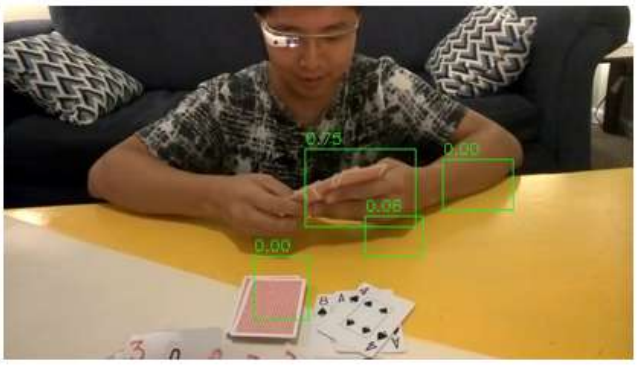
IoU = 0.01



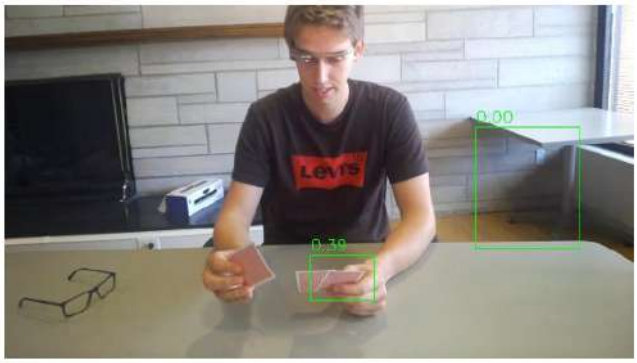
IoU = 0.00 – 0.60 – 0.00

IoU = 0.49

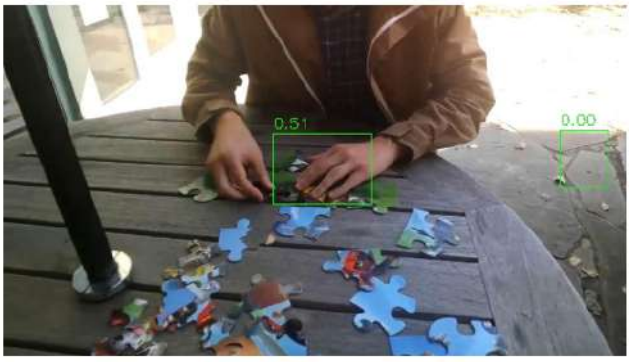
IoU = 0.46



IoU = 0.00 – 0.75 – 0.06 – 0.00



IoU = 0.39 – 0.00

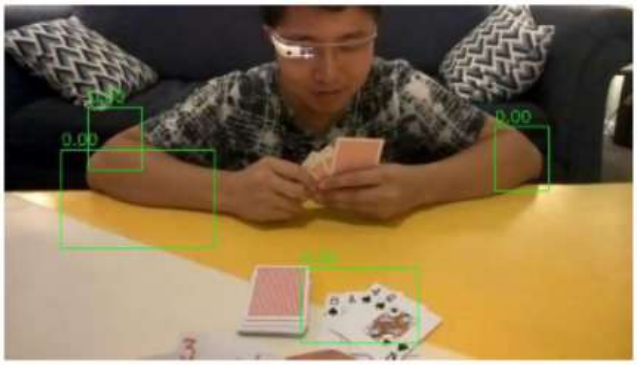


IoU = 0.51 – 0.00

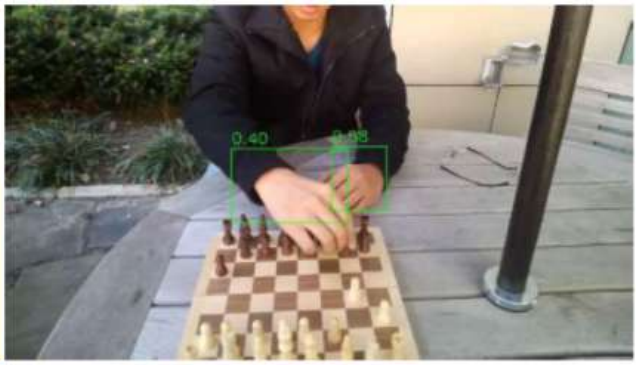
Experiment 3

For the third experiment, we combined the horizontal-hands detector of the first experiment (window size (112x72)) together with the vertical hands detector of the second experiment (window size (40x56)).

This led to more predicted hands with a better accuracy. Thus, we chose this detector as our final solution.

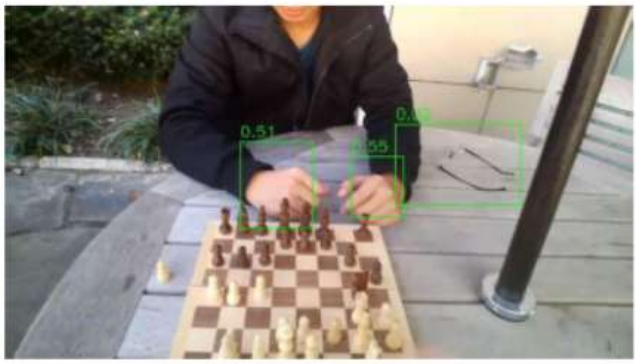


IoU = all 0.00

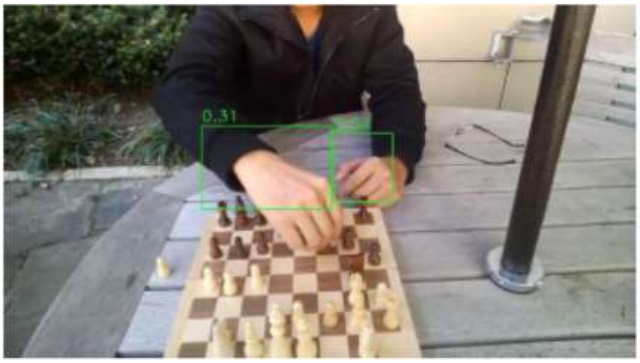


IoU = 0.40 – 0.58

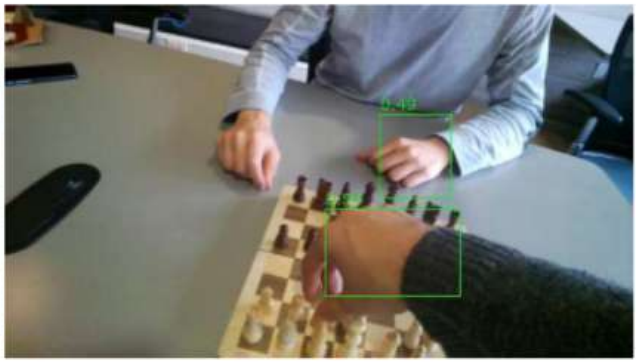
IoU = 0.43 – 0.81 - 0.00



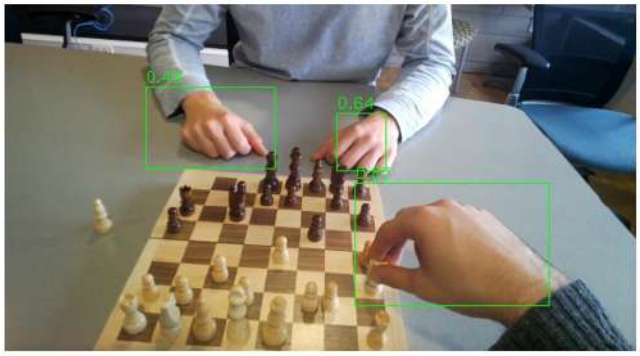
IoU = 0.51 – 0.55 – 0.02



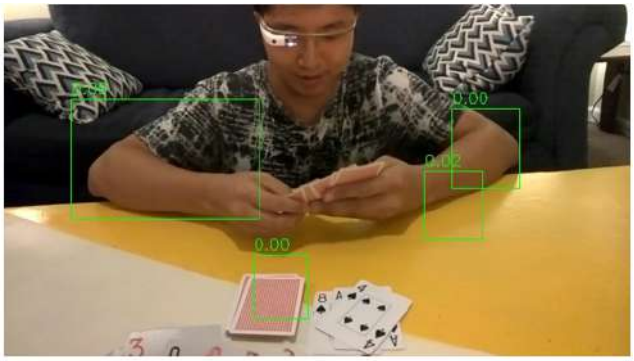
IoU = 0.31 – 0.54



IoU = 0.54 – 0.49

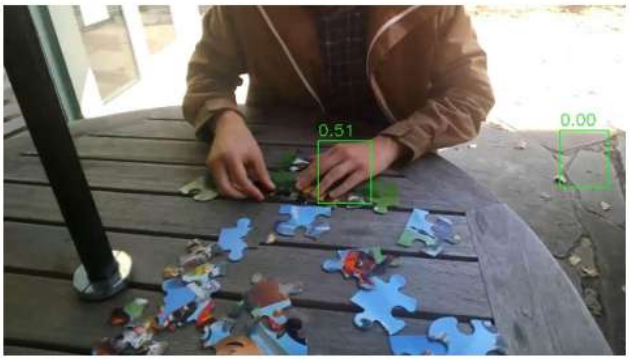


IoU = 0.48 – 0.64 – 0.65



IoU = 0.09 – 0.00 - 0.02 – 0.00

IoU = 0.19



IoU = 0.51 – 0.00

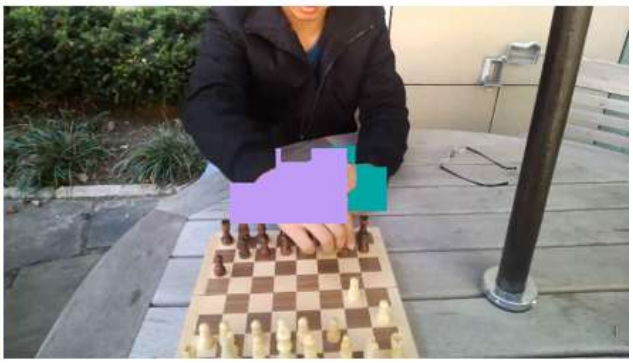
0.3.2 Segmentation results

We report the results we got by applying the different proposed segmentation methods. All the methods have been applied considering the predicted boxes returned by the detector used in the third experiment.

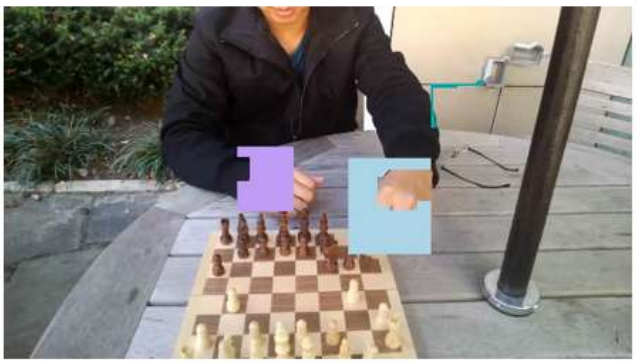
Contour-based method

The contour-based method resulted in poor results as shown below.

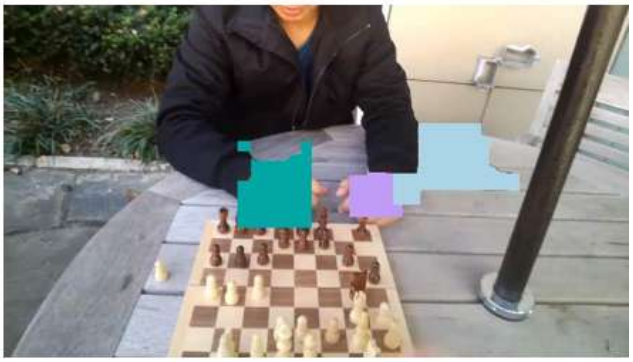
P_acc = 0.96



P_acc = 0.96

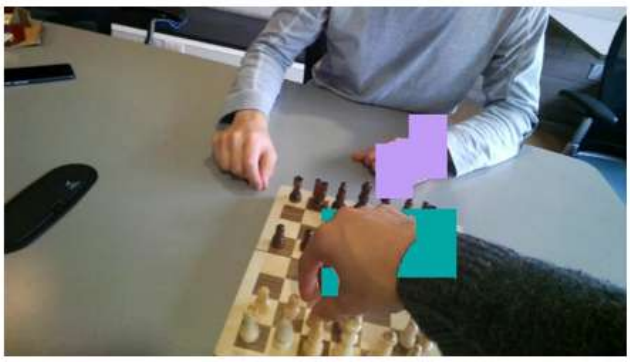


P_acc = 0.95

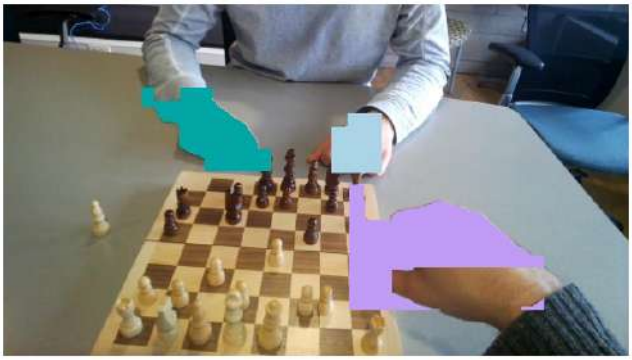


P_acc = 0.93

P_acc = 0.94

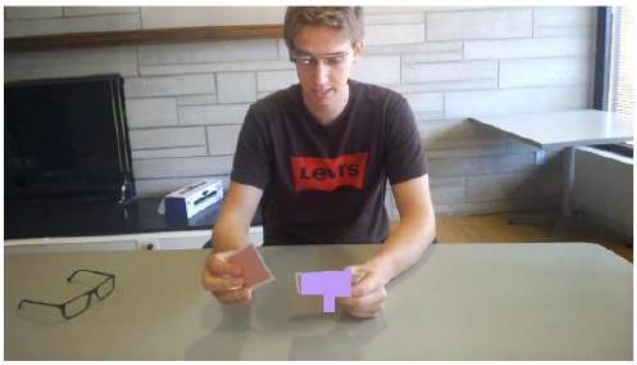


P_acc = 0.96



P_acc = 0.90

P_acc = 0.92

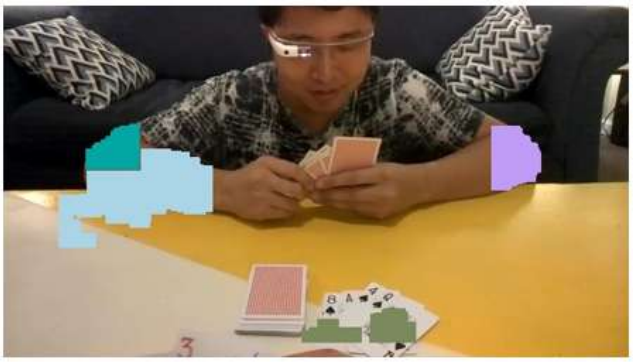


P_acc = 0.99

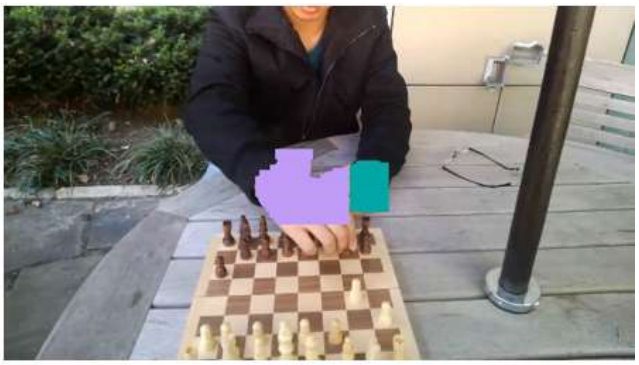
P_acc = 0.90

Colour thresholding-based method

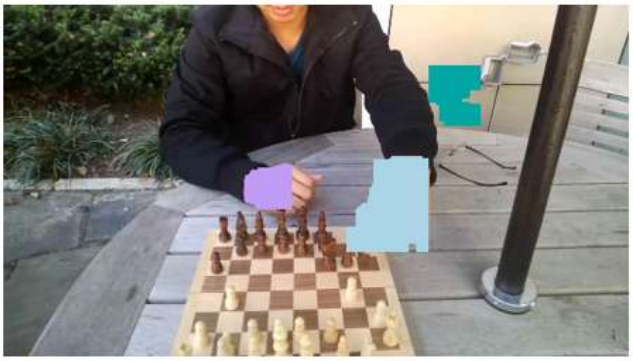
This method performs better than the previous one and it produces discrete results.



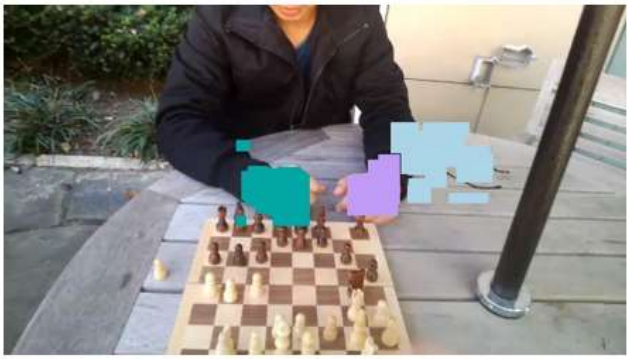
P_acc = 0.92



P_acc = 0.96



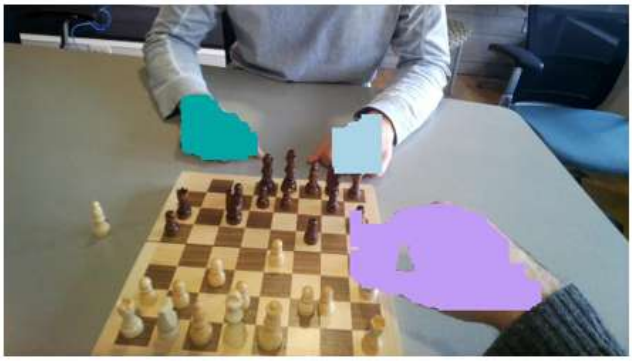
P_acc = 0.95



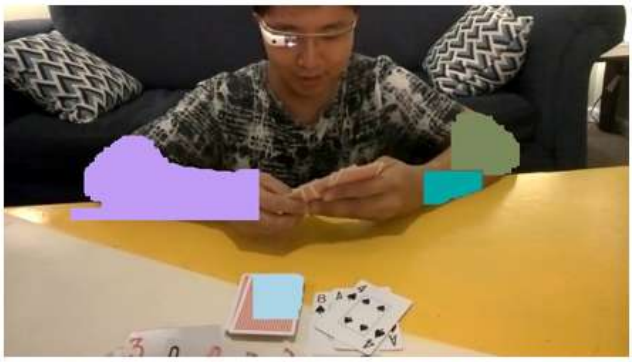
P_acc = 0.94

P_acc = 0.95

P_acc = 0.95



P_acc = 0.90



P_acc = 0.91

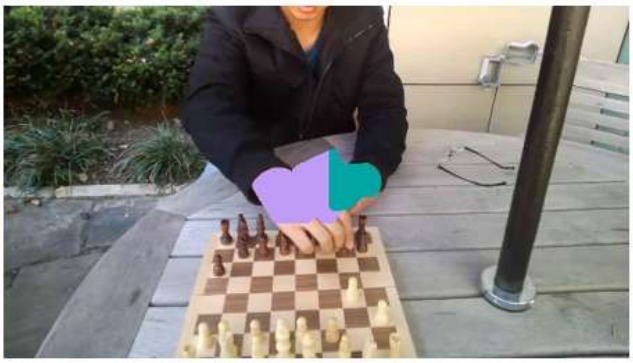
P_acc = 0.99

P_acc = 0.91

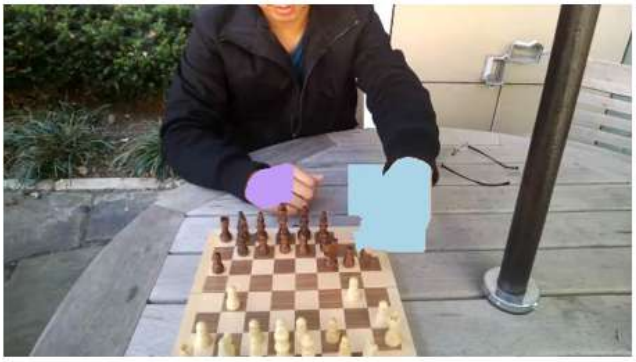
Grabcut method

The grabcut method produces results similar to the ones got by the colour thresholding method. However, the outcomes of the grabcut method appears to be more smoother. Moreover, this method can detect the background and avoid to detect background pixels. This statement can be verified by considering the third image and comparing it with the previous methods. We can see that the grabut method doesn't segment the wall pixels.

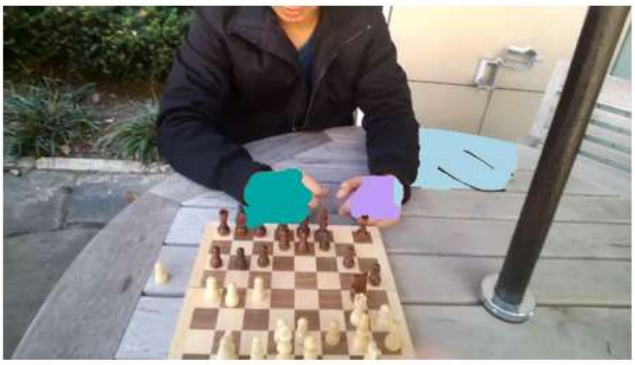
P_acc = 0.94



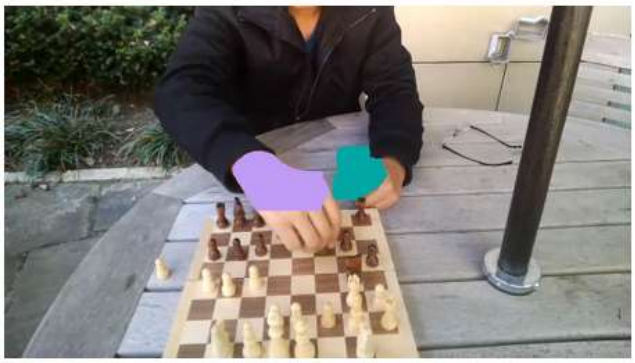
P_acc = 0.97



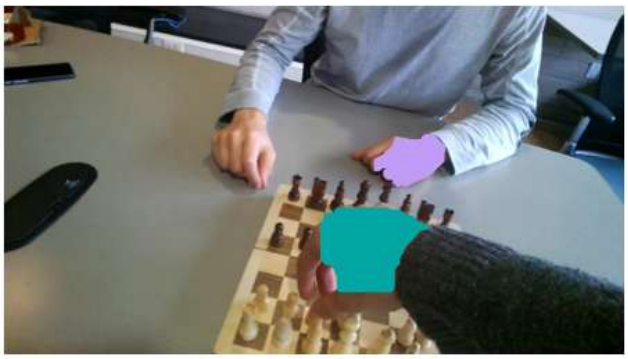
P_acc = 0.96



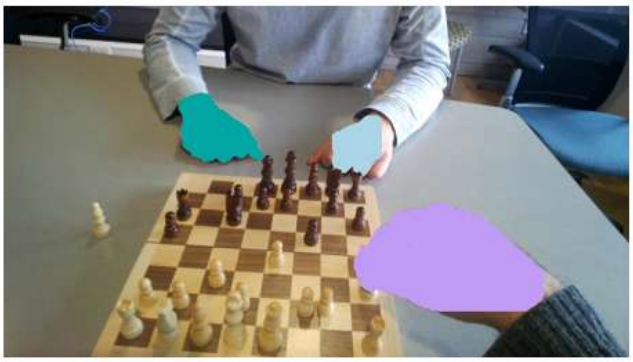
P_acc = 0.95



P_acc = 0.97



P_acc = 0.95



P_acc = 0.90

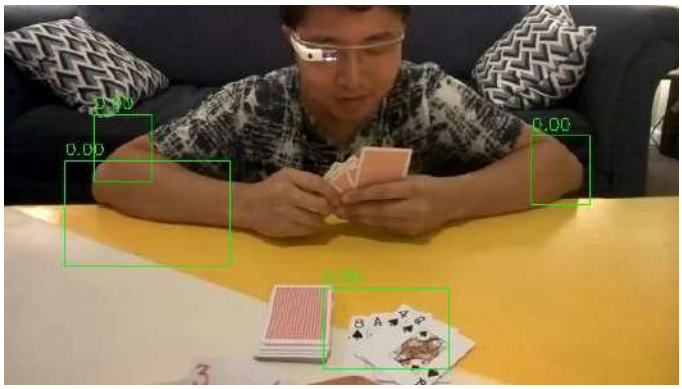
P_acc = 0.93

P_acc = 0.99

P_acc = 0.92

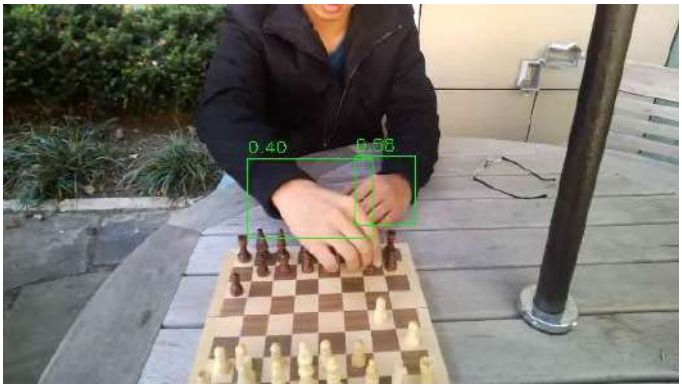
0.4 Final solution sum-up

We sum up here the outcomes of our final solution.



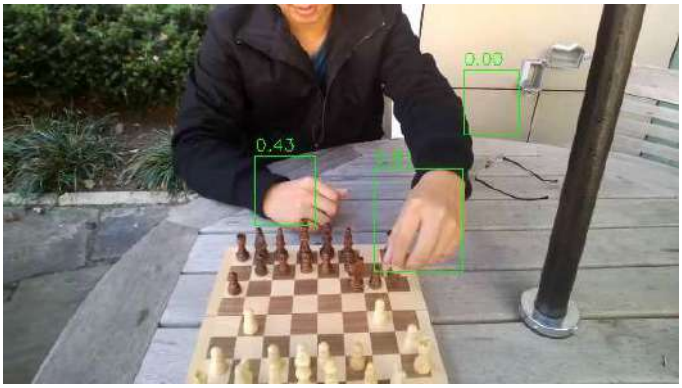
IoU = all 0.00

P_acc = 0.94

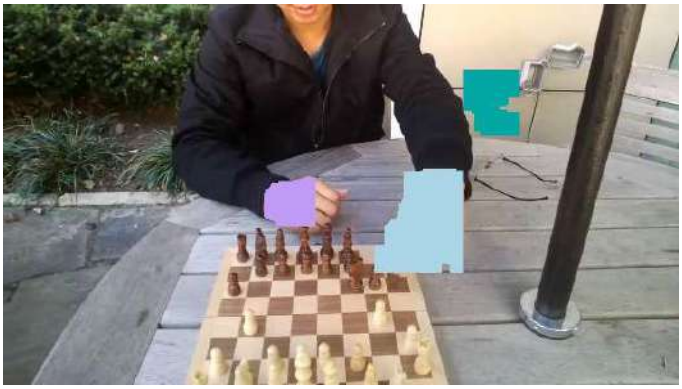


IoU = 0.40 - 0.58

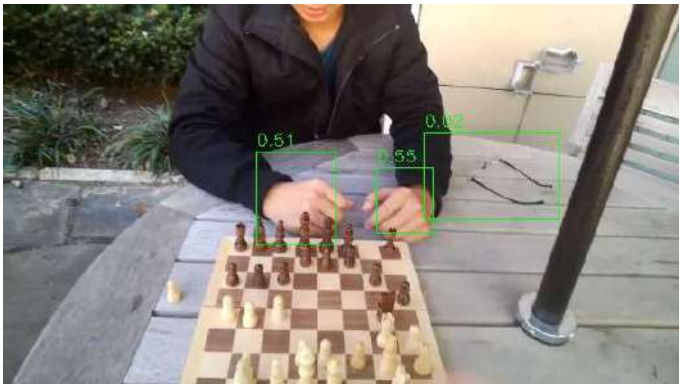
P_acc = 0.97



IoU = 0.43 - 0.81 - 0.00

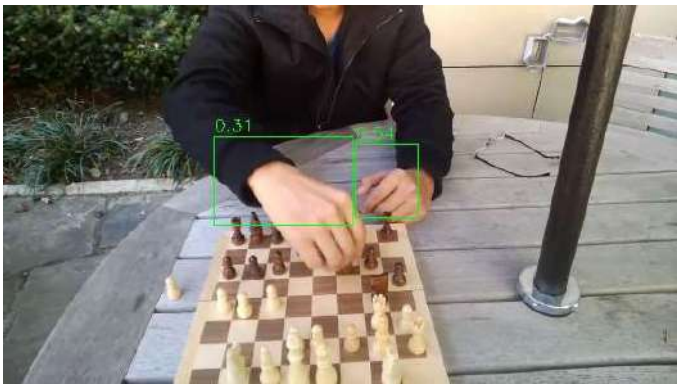


P_acc = 0.96



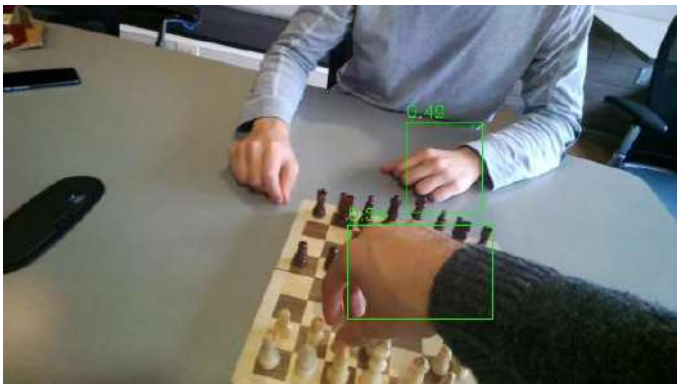
IoU = 0.51 - 0.55 - 0.02

P_acc = 0.95

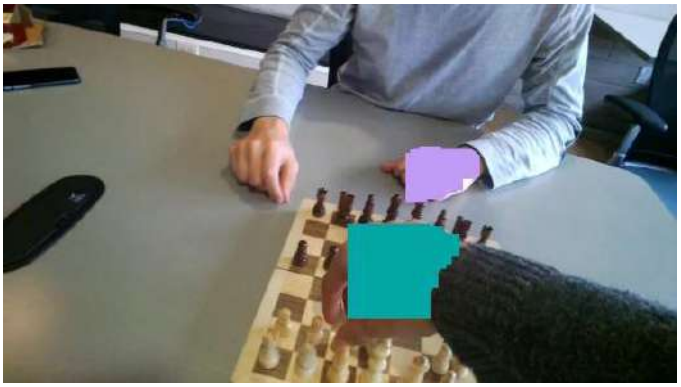


IoU = 0.31 - 0.54

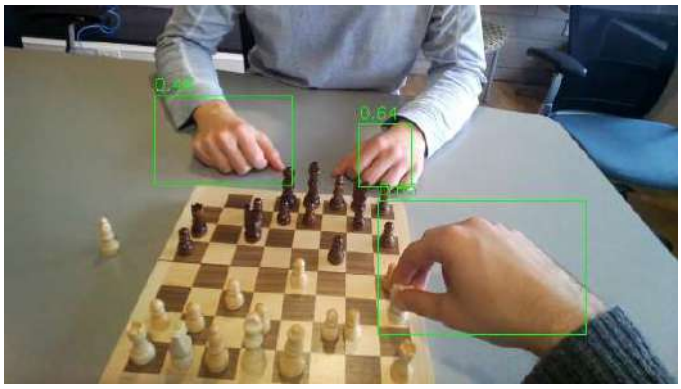
P_acc = 0.97



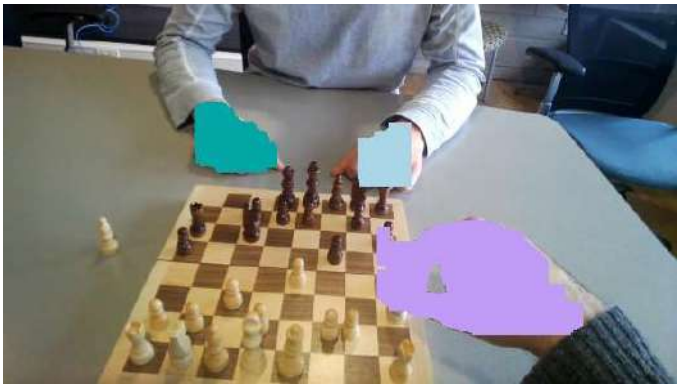
IoU = 0.54 - 0.49



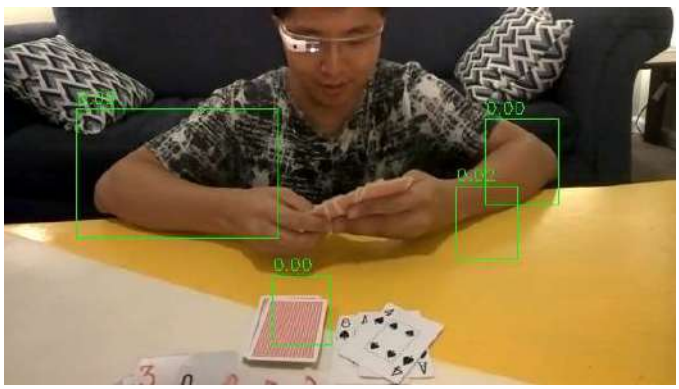
P_acc = 0.95



IoU = 0.48 - 0.64 - 0.65

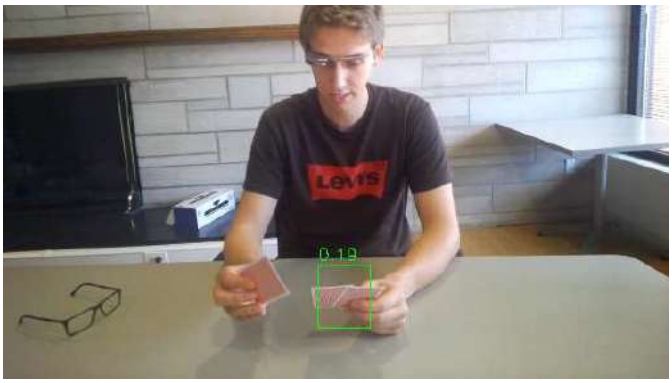


P_acc = 0.90



IoU = 0.09 - 0.00 - 0.02 - 0.00

P_acc = 0.93



IoU = 0.19

P_acc = 0.99

IoU = 0.51 - 0.00

P_acc = 0.92

0.5 Reference

1. https://public.roboflow.com/object-detection/hands

2. Human Skin Detection Using RGB, HSV and YCbCr Color Models.

   S. Kolkur1, D. Kalbande2, P. Shimpi2, C. Bapat2, and J. Jatakia2

3. Skin Detection using HSV color space

   V. A. OLIVEIRA, A. CONCI