

Summer Project: Hand Detection and Segmentation

Abstract

In this document we're going to explain how we approached the June/July project for the course of Computer Vision about hand detection and segmentation.

The main focus of our solution is based on machine learning techniques with the addition of some pre-processing phases made mostly with methods explained during the course.

1. Introduction

The task of object detection has had a growth in popularity in the last few years, especially with the spread of machine learning and deep learning models. So modern systems could need to detect any kind of real object or also body parts, like hands.

In this document we're going to explain how we dealt with this task. Starting detailing the techniques we used and the motivations that led us implementing them, then explaining the results we got on the benchmark dataset that was given to us.

During the course we didn't get so much into Deep Learning systems (which are defined as the state of art in terms of object detection), so we decided to rely on the combination of image processing methods we deepen the most during the lectures, such as for example morphological operators, feature detection/extraction and clustering with k-means and mean shift.

2. System Description

In this section we're presenting our choices for what concerns the structure and the components of the system we implemented to achieve the task's objective.

2.1 Training Dataset

To train and evaluate the model before the final benchmark of the system with the test images provided with the assignment of the project, as is good custom to do, we had to use some additional images to train some components we used in our solution.

For this purpose we used the "Hand-Over-Face" [1] dataset, paying attention to removing the ones that are in the image folder for the performance evaluation.

In addition, we must say that not all the images were used for training, mostly to try to keep the space dimension of the system as small as possible. This dataset was already provided with a ground truth, that we used essentially for creating two subsets: one containing the positive set, one containing the

negative set, respectively made by extracting what was in the bounding boxes defined in the ground truth, and deleting from the original images what was in the bounding boxes.

2.2 Image Pre-Processing

2.2.1 Skin segmentation

Searching for some possible approaches to the problem we came across a paper about skin segmentation in images. A phase of skin segmentation would've allowed us to make the system evaluate less regions and to reduce the amount of false positives. One of the negative aspects of this segmentation was that on one side it could delete from the image also some skin regions (i.e. shady regions), on the other it didn't always remove all the "non-skin" pixels.

This technique is based on constraints in YCbCr color space, more precisely on the Cb and Cr values. The ranges of values which were evaluated as skin were:

- $137 < Cr < 177$
- $77 < Cb < 127$
- $190 < Cb + 0.6 * Cr < 215$
- $105 < Cb < 140$ and $140 < Cr < 160$

We evaluated each pixel using an auxiliary image converted into YCbCr color space, if it didn't match previous criteria, we set it to RGB (0,0,0), that is black, into the output image.

2.2.2 Closing Operation

After the skin segmentation we tried to reduce the non-skin (and hopefully also skin but non-hand) regions that were segmented as skin even if they weren't. For this purpose we applied several times the closing operators that, as seen during the lectures, is the combination (in order) of the elementary morphological operators dilation and erosion.

The kernel used for this phase was the ellipsoid shaped one. This choice has been made considering that a "round" operator could have small influence into the parts of the hand on the image, that often are in some sort "rounded". On the other side this could more probably influence non-hand pixels, that often appeared as noise or with irregular shapes.

2.2.3 K-Means color segmentation

We decided also to add a color clustering phase into the image pre-processing. In this way, we could help the skin-segmentation reducing the probability to misclassify some skin pixels as non-skin. The clustering process used the found centroids as new colors for the image, getting a new one with more uniform regions.

2.2.4 Mean shift color segmentation

After k-means which was limiting for the number of clusters required, we also used mean-shift, testing it with different parameters we ended up selecting:

- $sp(\text{spatial window}) = 20$;
- $sr(\text{spatial color}) = 40$.



Example of mean shift clustered image with different parameters.

2.3 Detection

One of the goals of the project is to find the bounding boxes around hands, if there is any. So in a pixel-matrix oriented reasoning, we had to find a way to understand what “windows” of the images contained hands or part of hands.

Our initial idea was: slide the whole image, evaluate using some defined criteria if each window was or was not part of a hand and highlight them. In the end some kind of positional clustering method would've merged all the neighbor boxes found into a single one containing the whole hand.

2.3.1 SIFT features-histogram

The first strategy we tried was to use all the positive images (to recall: a set of images containing only hands) and extract the key points of each one. Then for each window of the image we would've extracted the keypoints. At this point we generated an histogram(one bin for each hand of the positive set) with the percentage of keypoints matching the region with the hand image corresponding to the image.

To decide if the window was part of a hand or not we computed the L2 norm and calculated the peak(bin with max value) of the histogram and compared them with thresholds.

Despite our high expectations about this solution, we tested it by varying the parameters but it was not as effective as we thought. So we decided to move on, searching for other solutions to the problem, but not going too far conceptually.

2.3.2 Histogram of Oriented Gradients

Searching for other approaches to the object detection problem we got into the histogram of oriented gradients. In a few words this method creates a descriptor for each window of a determined size in the

image, both for training an SVM classifier with training set (with positive and negative samples), and for the classification of validation/test images.

To implement this solution we firstly adapted the code presented in [2], basically to train the classifier. Then we created a class called HOGwSVM that loads the trained detector and then uses it to detect hands on images.

The recognition is performed using pre-processed images. The reason behind this choice is because we saw that this could help for what concerns the precision of the box's location, so basically those elements represent only skin classified pixels in which several morphological operations have been executed.

2.3.3 Bag of Words-SVM

This method was the first we tried for the detection. We tried in several different ways to implement Bag of Words (initially some implementation details were not clear to us), always complementing them with what we thought should've been done for completing the process.

This first version with SVM was implemented as follows: first we extracted the key points from all the images in the training set and we divided them into positive and negative labels(hand and non-hand). Then we tested 2 strategies: first we extracted a set of keypoints using k-means for each class and then we passed them to SVM for training, the other method just skipped the extraction of key points with k-means. Both of them gave very poor results on the validation set and we gave up after many times of train with different parameters and different normalizations.

2.3.4 Bag of Words-Kmeans

After the unsuccessful approach with SVM we decided to implement the BOW method as shown during lectures: we extracted a set of features from all images in the training set and we divided them into positive and negative labels; then we used k-means to the entire set of key points and for each centroid we calculated the ratio between all positive and negative points belonging to it. At the end we keep as positive key points only points with a score greater than a threshold(th1) and as negative all the points with a score less than another threshold(th2), with $th1 > th2$. $th1 = 0.38$ and $th2 = 0.17$.

To find hands in an image we used a sliding window approach and at each iteration we calculate an histogram with positive and negative keypoints and we keep only images with the sum of values on the positive key points greater than a threshold th.

2.3.5 Bounding Boxes Processing

During the training process, we found out that often our detectors found lots of boxes that could belong to hands, or too big that encapsulated more than one.

Firstly we decided to exclude all rectangles that contained less skin pixels than a defined threshold, then we decided to split all the rectangles in small squares and apply this criteria to those.

So we needed a way to merge this amount of squares. The first thing we thought was clustering them by position, for example with k-means. As said before, and considering that not all the images have the same amount of hands, this was a limited approach for this task.

Then we discovered that OpenCV has a clustering method that could help us for this task, that is `cv::Partition`. So we partitioned the rectangles basing on their intersection over a defined threshold[X].

One thing that we found interesting is that representing those small squares left we got a primitive segmentation, at least under the visual aspect.

2.4 Segmentation

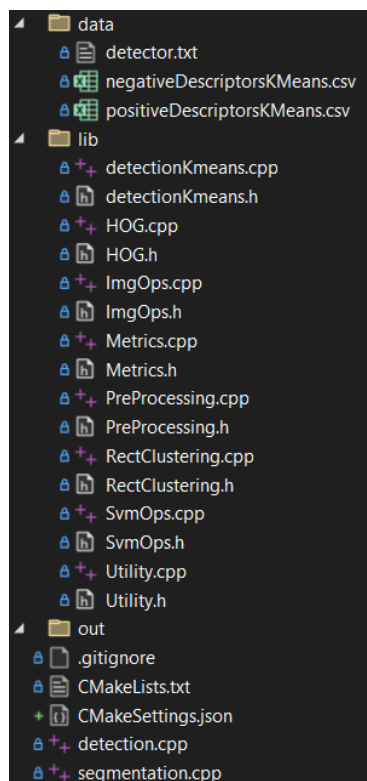
Considering that the boxes we got from “mini-squares clustering” were all containing skin pixels, also considering the time we got, we decided to use them as hand pixels, and so for the segmentation module. We computed the area inside each bounding box and coloured all pixels that were classified as skin. In this phase we used again a pre-processed image with the technique that allowed us to improve the accuracy of the segmentation.

2.4 System’s Pipeline

Each module starts adding to the command for executing it a path to the dataset folder. We decided to realize a system that simply executes the required tasks as required. The only things users can do are choosing the dataset directory and watching the processed images and their performances at the end of the execution.

The systems then goes through the following steps:

1. Starts the execution with as argument the path to the dataset folder containing images, masks and annotations.
2. Load images from the given path
3. Load pre-trained detectors, using files in the “data” folder.
4. Starts a cycle through the images:
 - a. Pre-Process of the image
 - b. Get meaningful rectangles from both detectors
 - c. Split into smaller rectangles
 - d. Cluster mini squares rectangles and remove smallest rectangles
5. Represent output images, each one with the predicted hand squares or the segmented hands with a different color each one.



Organization of the folders with the code.

3. Output of the System and Performance Evaluation





3.1 Single image performance

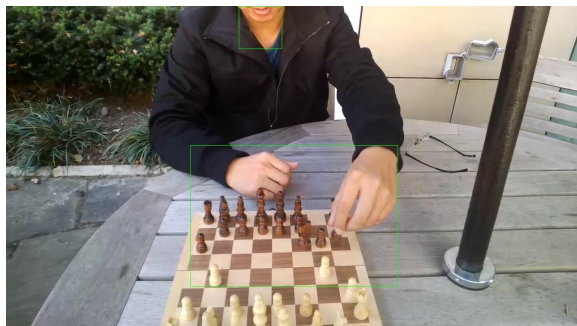
Single image performance was evaluated for the two tasks: detection and segmentation.

For the detection part we used the IOU(intersection over union) metric on bounding boxes for each true and predicted bounding box.

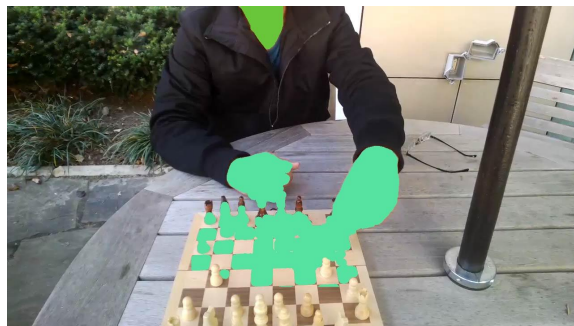
For the segmentation part the pixel accuracy calculated normally could return a high score even if the segmentation is not good since the negative(non hand) class is usually much higher than the positive(hand). Instead for pixel accuracy we used the method of IOU calculating the ratio between area of intersection and area of union for both classes.

In some images the hands are well detected while for other images there may be some missing hands or even no hands at all.

Detection	Segmentation
 IOU:0.166499	 Correctly classified as hands:0.248655 Correctly classified as background:0.866463 Pixel accuracy:0.557559
 IOU:0.267566	 Correctly classified as hands:0.460943 Correctly classified as background:0.958471 Pixel accuracy:0.709707



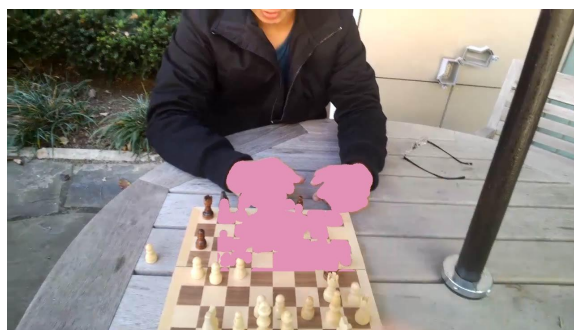
IOU:0.218975



Correctly classified as hands:0.379956
Correctly classified as background:0.943252
Pixel accuracy:0.661604



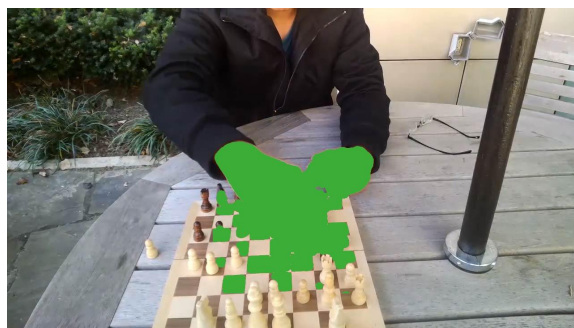
IOU:0.27009



Correctly classified as hands:0.365786
Correctly classified as background:0.956682
Pixel accuracy:0.661234



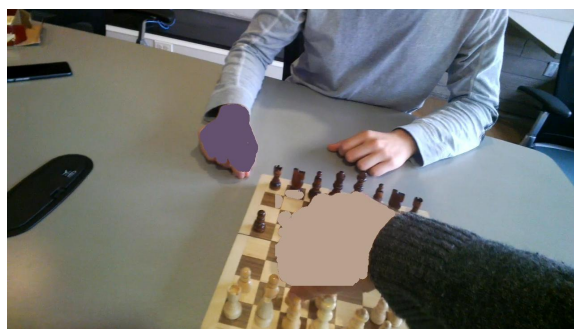
IOU:0.308185



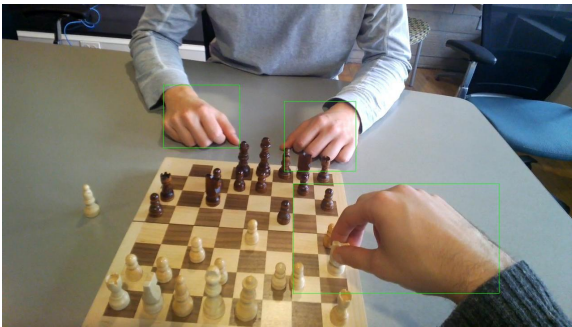

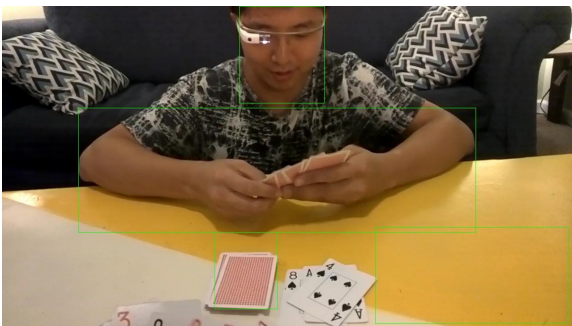

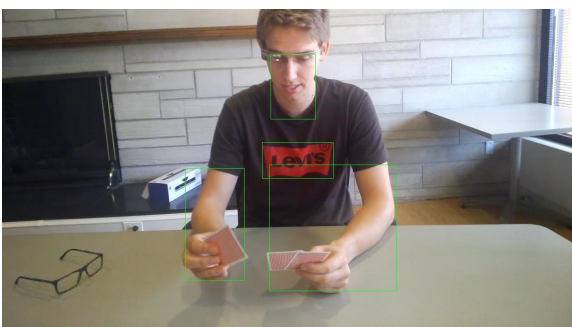

Correctly classified as hands:0.502349
Correctly classified as background:0.957329
Pixel accuracy:0.729839

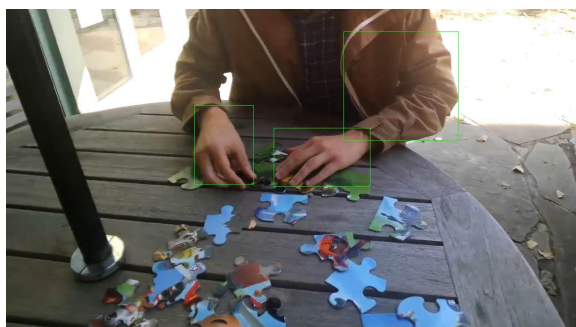


IOU:0.387736

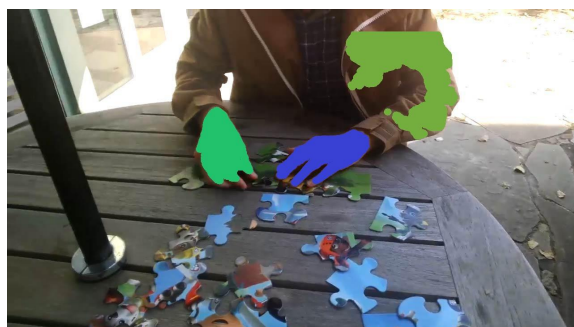


Correctly classified as hands:0.662106

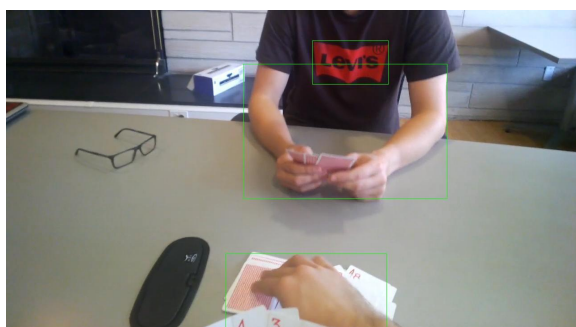
	<p>Correctly classified as background:0.971388 Pixel accuracy:0.816747</p>
 <p>IOU:0.416841</p>	 <p>Correctly classified as hands:0.69552 Correctly classified as background:0.962084 Pixel accuracy:0.828802</p>
 <p>IOU:0.126911</p>	 <p>Correctly classified as hands:0.181295 Correctly classified as background:0.788129 Pixel accuracy:0.484712</p>
 <p>IOU:0.130051</p>	 <p>Correctly classified as hands:0.197583 Correctly classified as background:0.942651 Pixel accuracy:0.570117</p>



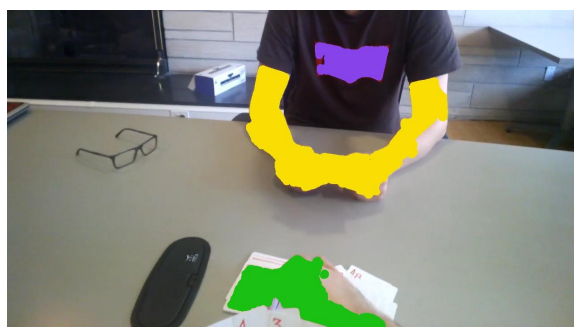
IOU:0.282336



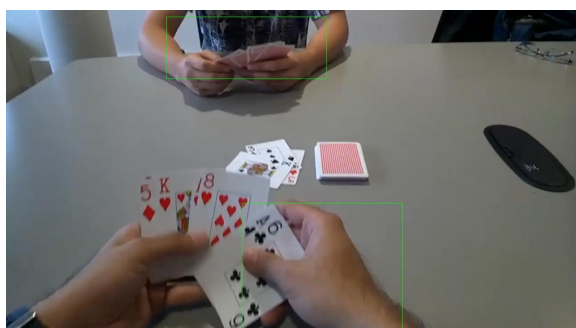
Correctly classified as hands:0.387332
Correctly classified as background:0.954045
Pixel accuracy:0.670689



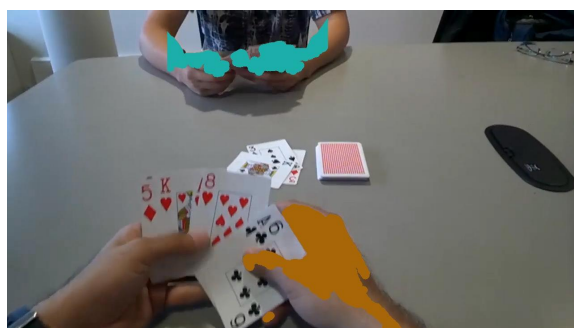
IOU:0.246528



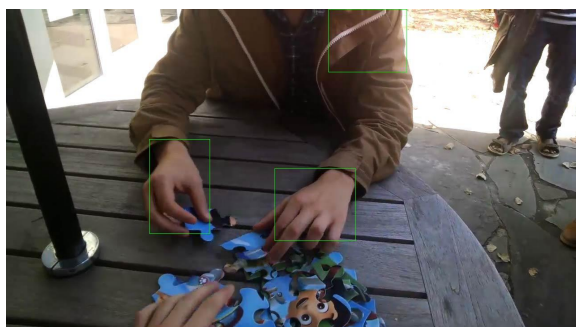
Correctly classified as hands:0.366264
Correctly classified as background:0.930629
Pixel accuracy:0.648446



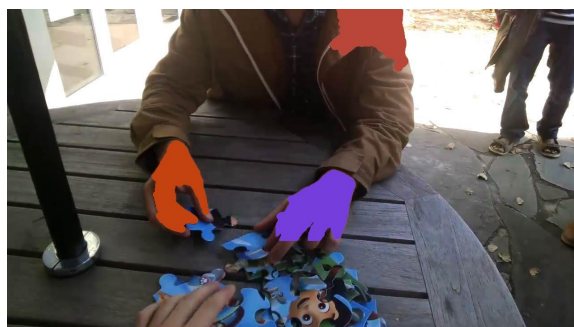
IOU:0.291964



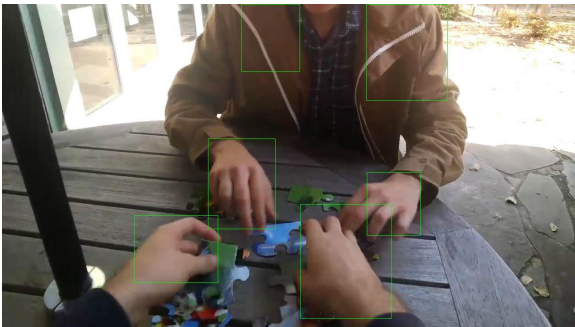
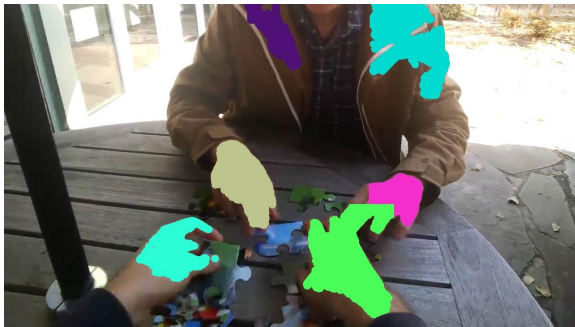
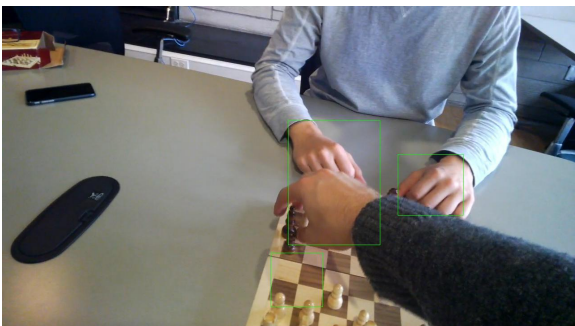
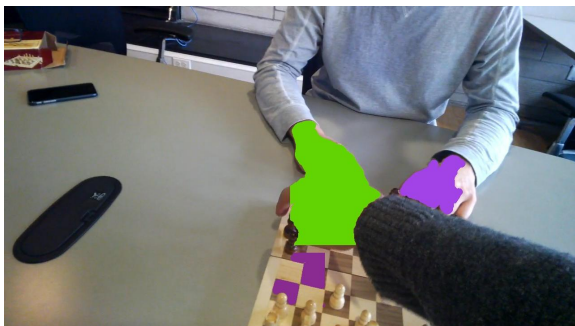
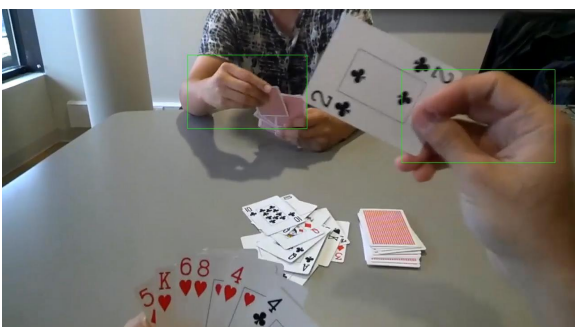
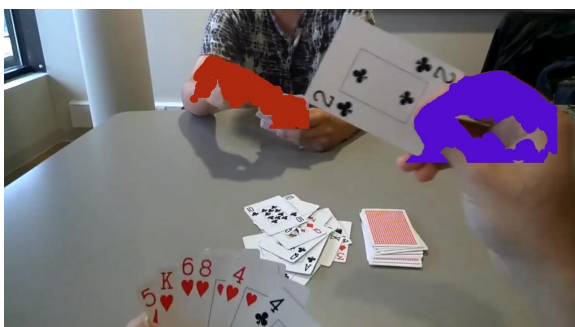
Correctly classified as hands:0.290623
Correctly classified as background:0.892034
Pixel accuracy:0.591329



IOU:0.289955



Correctly classified as hands:0.388104

	<p>Correctly classified as background:0.94316 Pixel accuracy:0.665632</p>
 <p>IOU:0.374195</p>	 <p>Correctly classified as hands:0.488058 Correctly classified as background:0.919648 Pixel accuracy:0.703853</p>
 <p>IOU:0.406525</p>	 <p>Correctly classified as hands:0.666153 Correctly classified as background:0.977411 Pixel accuracy:0.821782</p>
 <p>IOU:0.26673</p>	 <p>Correctly classified as hands:0.349144 Correctly classified as background:0.882933 Pixel accuracy:0.616039</p>



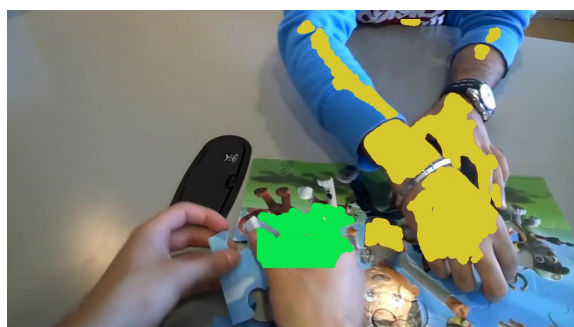
IOU:0.284022



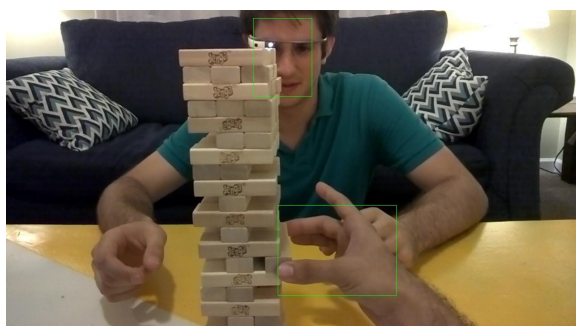
Correctly classified as hands:0.344259
Correctly classified as background:0.932314
Pixel accuracy:0.638286



IOU:0.241869



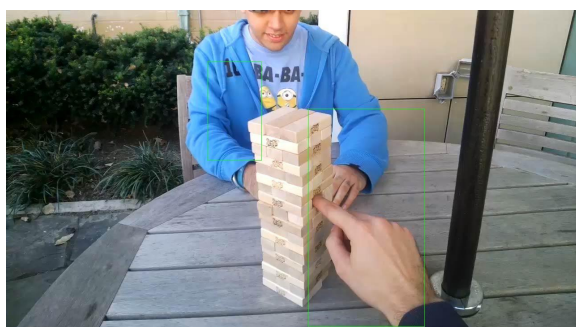
Correctly classified as hands:0.295739
Correctly classified as background:0.818895
Pixel accuracy:0.557317



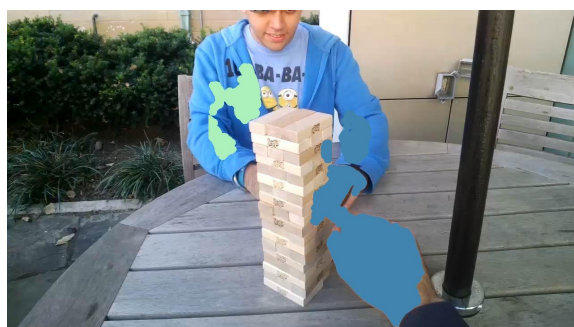
IOU:0.297137



Correctly classified as hands:0.29258
Correctly classified as background:0.929724
Pixel accuracy:0.611152



IOU:0.310056



Correctly classified as hands:0.553196

	Correctly classified as background:0.961967 Pixel accuracy:0.757581
--	--

3.2 Overall System performance

Calculating the average of the considered values we obtain the following:

<i>Average IOU of the System</i>	0.279209
<i>Average correctly classified as hands per image</i>	0.405782
<i>Average correctly classified as background per image</i>	0.92446
<i>Average Pixel Accuracy</i>	0.665121

3.3 Considerations on performance

Considering that no deep learning methods were used in the final solution the overall system performance was not too bad; we based our bounding boxes on two different methods for detection plus the segmentation on the image which improved the final results of bounding boxes by a lot. We also tried to train a SVM using SIFT features extracted from the images but since the results on the validation data were not good we didn't add it to the final model.

Observing how the system behaves on the test dataset we can certainly say that visually it performs discreetly. Most of the time it catches all hands. On the other side there are the requirements of the project, like that each hand should've had its own bounding box, and the system performance, that are quite negative from our point of view, specially the IOU. For this last aspect we must consider that sometimes the predicted boxes were too big, and on the other hand there were some false positives and false negatives too which strongly influenced the detection metric.

4. Conclusions

In conclusion the implementation of the system we realized was not as good as we expected, but indeed had some positive aspects to consider. For example, the method detects almost all the hands. The segmentation is often quite satisfying, considering it's not a proper "smart" method, but surely needs some improvements.

Some ideas to improve the performance of the system could be for example use a wider dataset for training the detectors, for sure implementing some deep learning systems to get better accuracy in some phases. One idea to split hands that were in the same bounding box could be to apply an edge mask with very thick edges, so that using the clustering of mini-rectangles explained before we can separate those regions.

5. Members' Contributions

5.1 Working hours

Considering approximately 3 or 4 hours of research in the first days of July and 8/10 hours a day from 19 to 28 July for effective work it took around 90/100 hours each.

We could've reduced the amount of work relying on deep learning, so for some aspects it depends on the time someone has/wants to spend on it.

5.2 Ideas and Implementation

In general both of us contributed in all the phases of the realization of the project, from idea brainstorming, to code implementation and for writing the report.

But having to list where each member contributed the most we have:

- Alberto:
 - Bag of Words with K-means
 - Basic operations for the system like ground truth reading from file, positive and negative dataset generation and feature extraction
 - System testing and parameters fitting
 - Skin Segmentation method
 - Metrics calculation functions
 - Report Writing
- Riccardo
 - Bag of Words with SIFT
 - HOG detector
 - Image Pre-Processing
 - Basic operations for the system, like rectangle resizing, segmentation function
 - Rectangles clustering
 - System testing and parameters fitting
 - Report Writing

6. References

- [1] "Image Segmentation for Skin Detection." n.d. Journal of Southwest Jiaotong University.
<https://www.jsju.org/index.php/journal/article/view/479>.
- [2] https://docs.opencv.org/3.4/d0/df8/samples_2cpp_2train_HOG_8cpp-example.html#a33