Universitatea din București
Virtute et Sapientia

Facultatea de Matematică și Informatică
Universitatea din București

Homework 2– Concepts and Applications in Computer Vision

Student: Sebastian-Ioan ENE
Class 352 - Bachelor Computer Science

22 January 2023

1 Short introduction

The project requires sorting out a theme of facial recognition of cartoons from the "Life with Louie" series. This homework is to be 50% of the grade for the Concepts and Applications in Computer Vision course.

The wish is to apply techniques presented in the course and laboratory ("sliding window" method), and not pre-training of deep-learning models such as YOLO or Faster-RCNN.

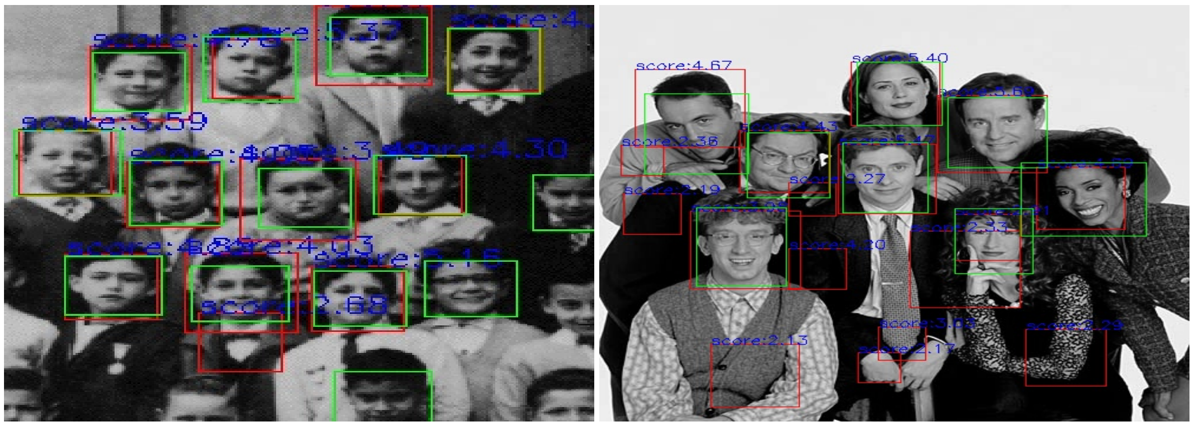
2 The start

I started this project by implementing the sliding window method for the project of the previous year, the one with facial detection for humans. Initially I had an average precision (AP) of 4%, but changing some parameters I reached higher percentages, but after implementing the sliding window method, I had tens of percentages however the false detections appeared more often.

3 Formation of a first idea

It became quite clear to me that the images from the cartoons varied a lot, more so than human faces, because of the numerous facial expressions intended to impress or capture the attention of children. Therefore, I considered that if I reach 70-80% accuracy with the human face detection project of last year, then I could be prepared to switch to the current project.

I give examples of images that I obtained from face detection using sliding window method. The performance was not exceptional, but the intent was to have a base for this years' project on which the mark is given, and not double the efforts for the grade. I observed some false positive detections, even after applying hard negatives filtration. This fact is understandable, because the sliding window is a bit outdated, [N. Dalal and B. Triggs. Histogram of Oriented Gradients for Human Detection. In CVPR, 2005] and there are more optimal techniques nowadays such as neural networks (YOLO/ Faster-RCNN) but they are not allowed to be used in this academic project.



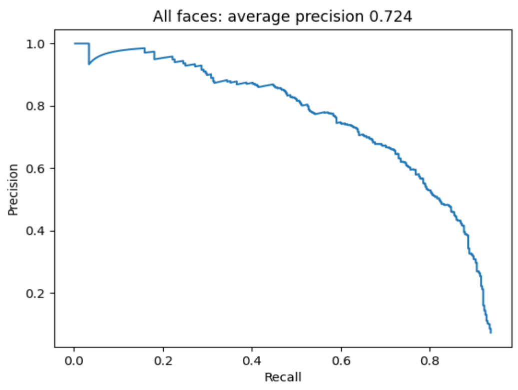
(a) Facial detection homework from last year (b) Facial detection homework from last year

Figure 1: Facial detection homework from last year

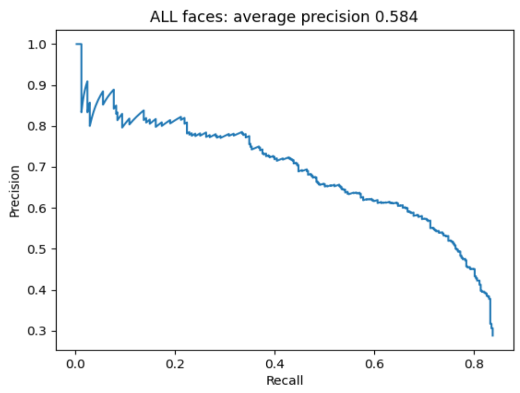
4 Relevant images from the project

4.1 Task 1

In the two figures below, there are shown the AP scores obtained from the PDF given by the professors as a benchmark, and compared with what I obtained. In the images, I used a white rectangle to show where the face of a character should be detected. Afterwards there are a series of face detection examples for characters randomly chosen.

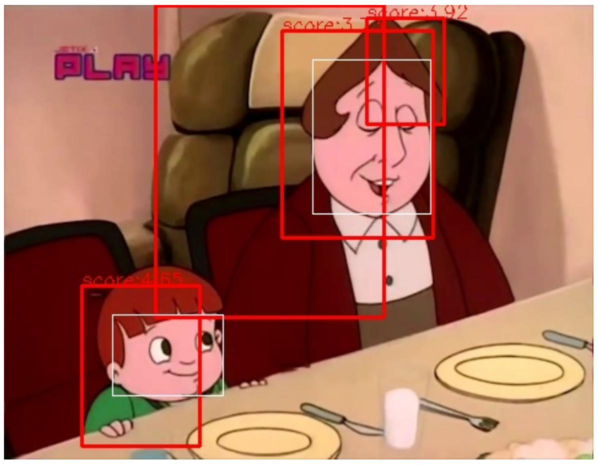


(a) AP of PDF Task 1 from professors as a benchmark

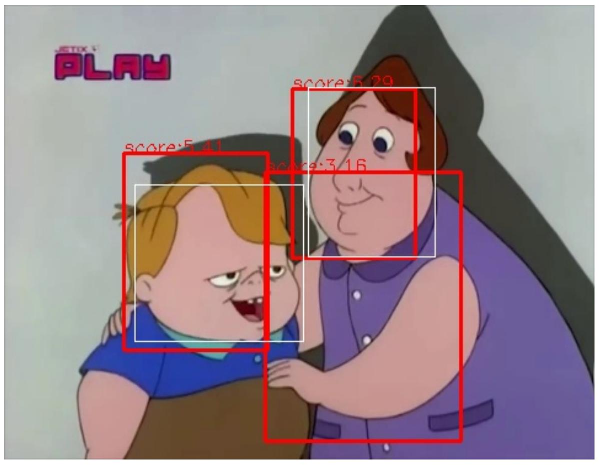


(b) AP obtained by me for Task 1

Figure 2: My AP result for Task 1 compared to the benchmark one from the PDF of professors

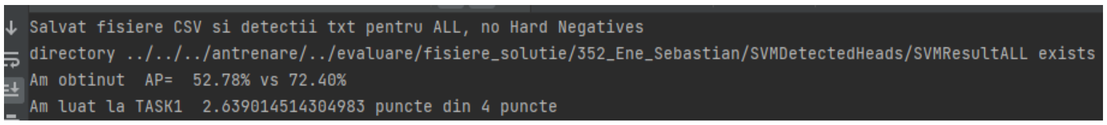


(a) Facial detection of characters



(b) Facial detection of characters

Figure 3: Facial detection of characters

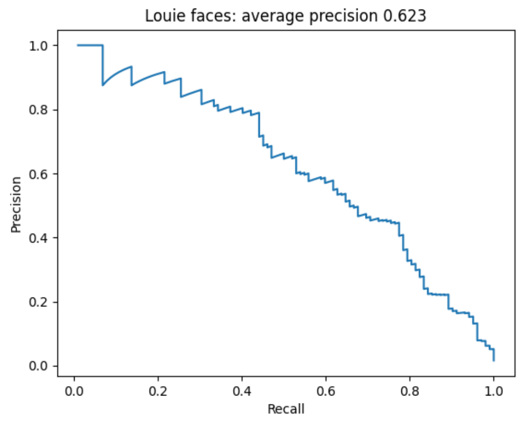


```
Salvat fisiere CSV si detectii txt pentru ALL, no Hard Negatives
directory ../../../antrenare/../evaluare/fisiere_solutie/352_Ene_Sebastian/SVMDetectedHeads/SVMResultALL exists
Am obtinut  AP=  52.78% vs 72.40%
Am luat la TASK1  2.639014514304983 puncte din 4 puncte
```

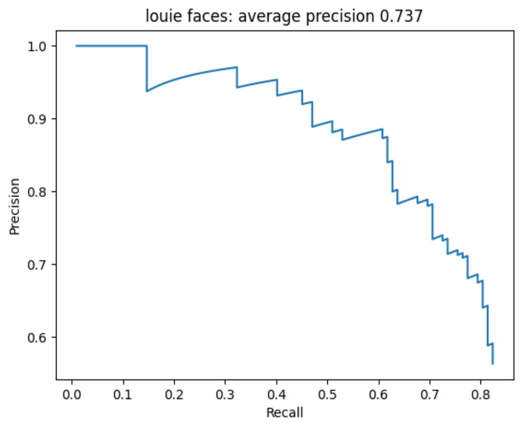
Figure 4: Automatic calculation of Task1 score for the validation set

4.2 Task 2

In the sets of figures below, it is shown the AP scores of the PDF with the requirements of Task 2 versus the one I obtained. In the images, I used a narrow white rectangle on the image area where there is a character face to be detected. There are afterwards examples of facial recognition for Louie, Andy, Ora and Tommy.

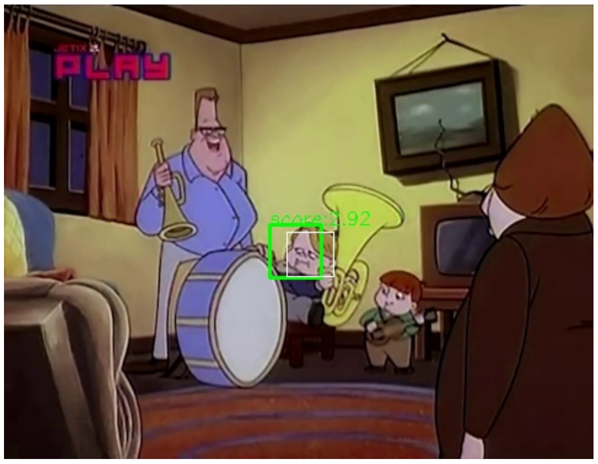


(a) AP Louie of PDF Task 2

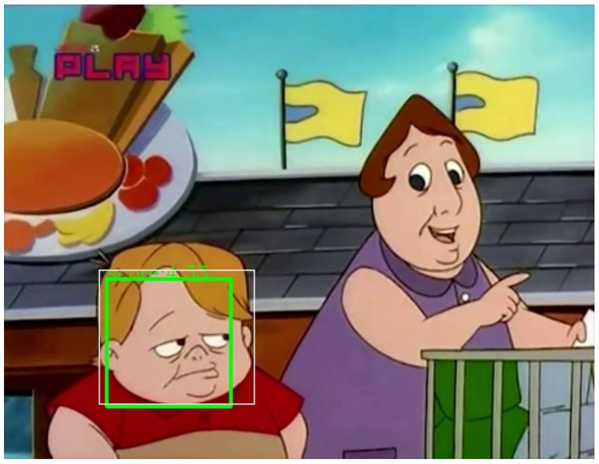


(b) AP Louie obtained by me

Figure 5: AP Louie my result versus the PDF from professors with the requirements

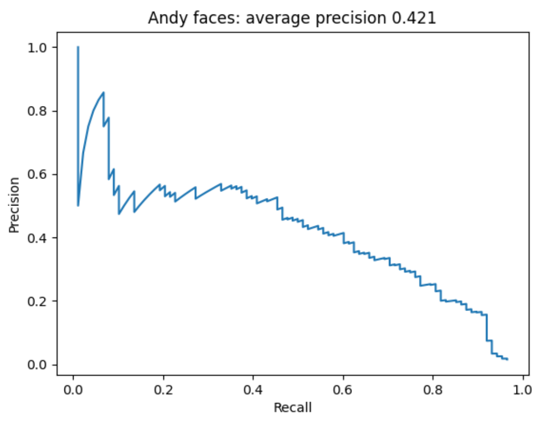


(a) Facial recognition Louie

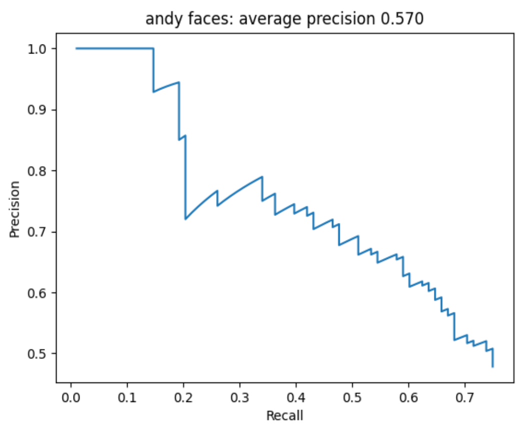


(b) Facial recognition Louie

Figure 6: Facial recognition Louie

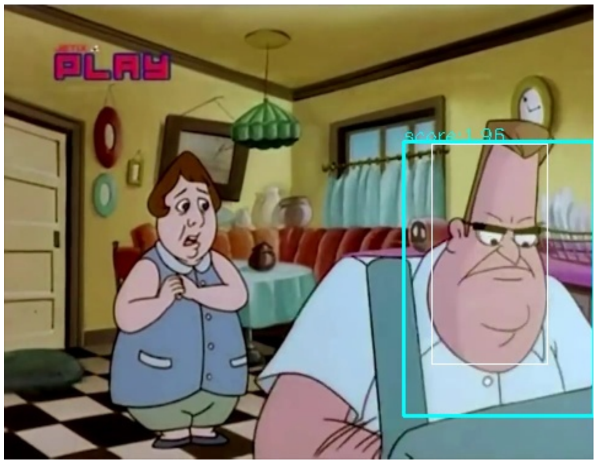


(a) AP Andy of PDF Task 2

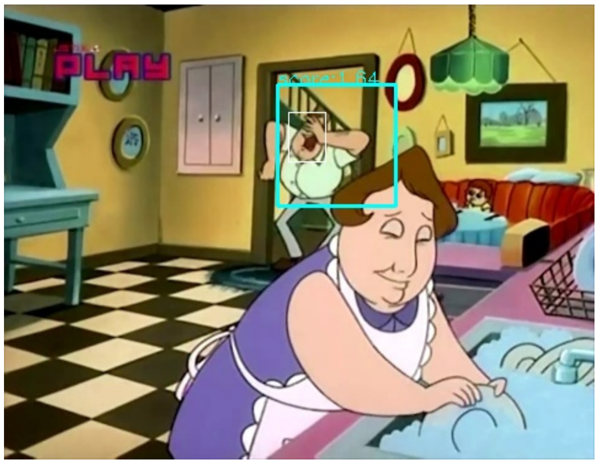


(b) AP Andy obtained by me

Figure 7: AP Andy my result versus the PDF one

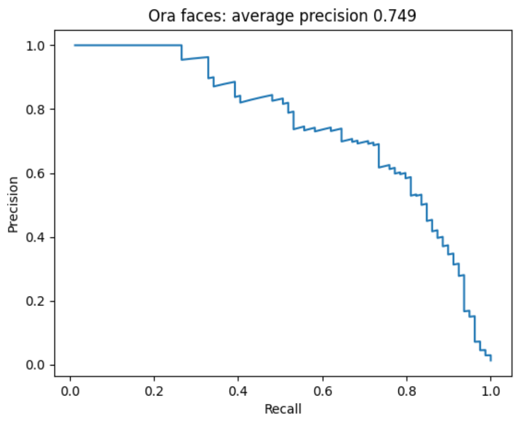


(a) Facial recognition Andy

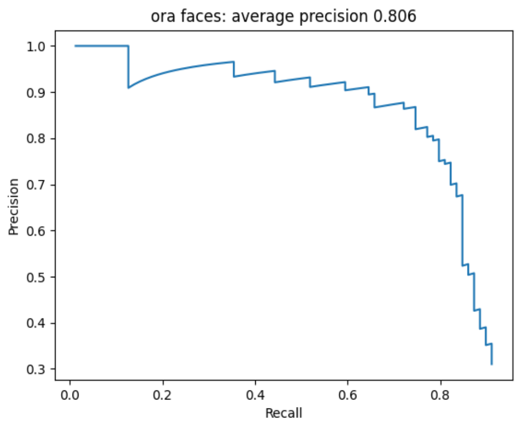


(b) Facial recognition Andy

Figure 8: Facial recognition Andy

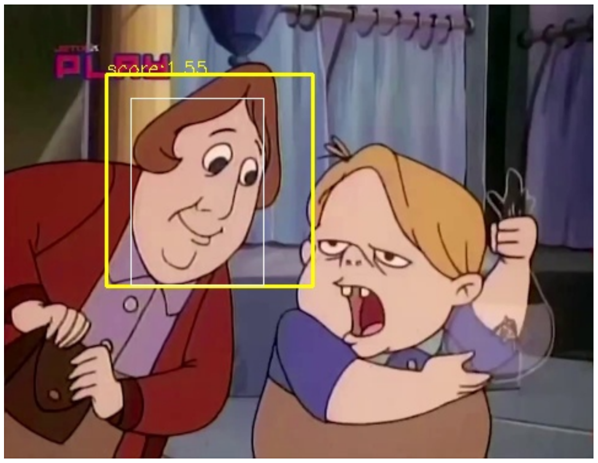


(a) AP Ora of PDF Task 2

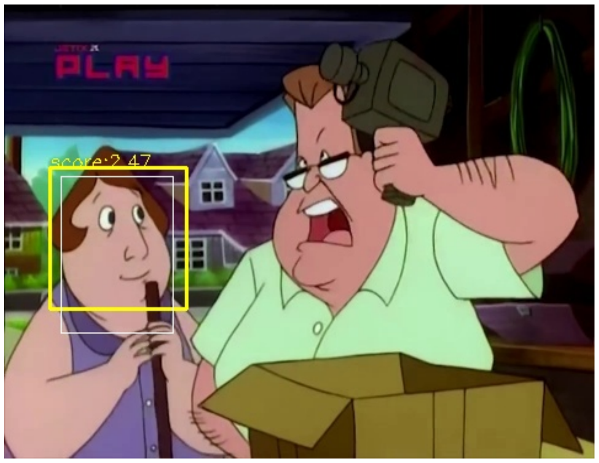


(b) AP Ora obtained by me

Figure 9: AP Ora my result versus the one from the PDF professors

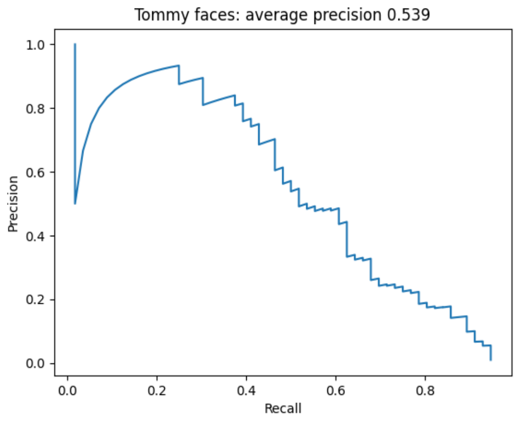


(a) Facial recognition Ora

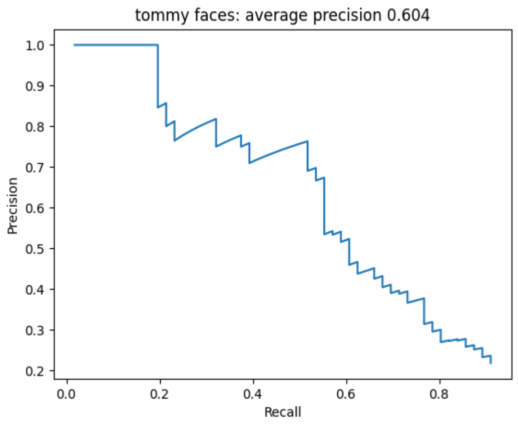


(b) Facial recognition Ora

Figure 10: Facial recognition Ora

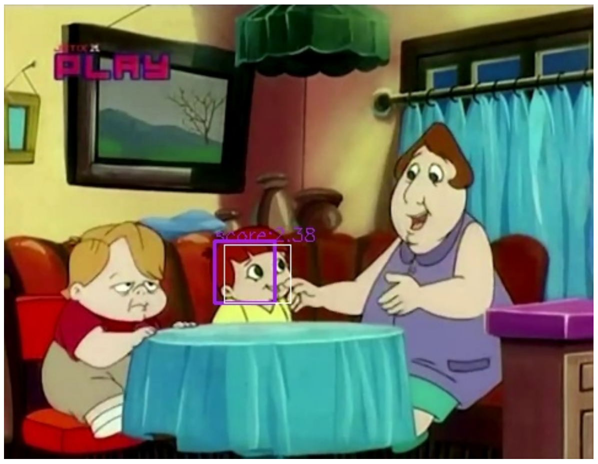


(a) AP Tommy of PDF Task 2

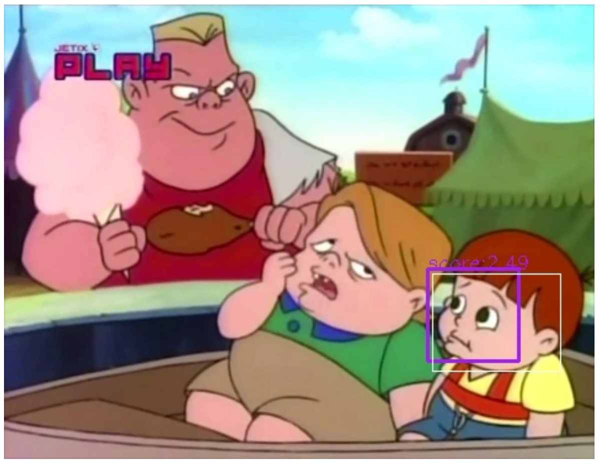


(b) AP Tommy obtained by me

Figure 11: AP Tommy my result versus the one of the PDF from professors

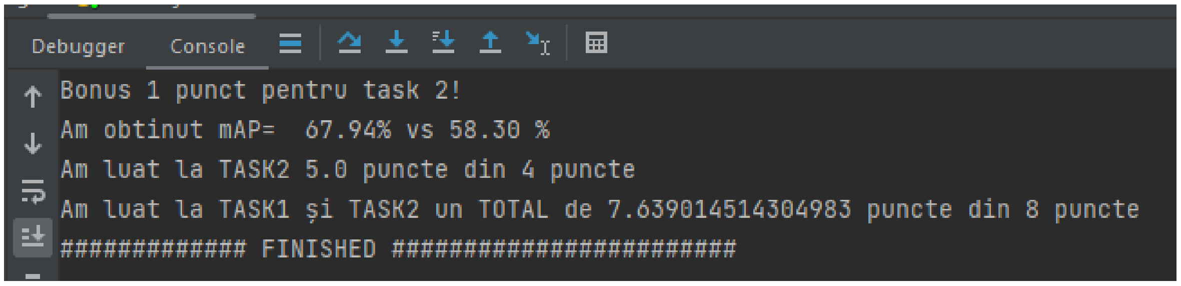


(a) Facial recognition Tommy



(b) Facial recognition Tommy

Figure 12: Facial recognition Tommy



```
Bonus 1 punct pentru task 2!
Am obtinut mAP=  67.94% vs 58.30 %
Am luat la TASK2 5.0 puncte din 4 puncte
Am luat la TASK1 și TASK2 un TOTAL de 7.639014514304983 puncte din 8 puncte
############# FINISHED ########################
```

Figure 13: Automatic score calculation Task 2 and total points on the validation set

5 Details/ the working strategy

5.1 Timing

This project seemed so complex to me that initially I did not know if I should start it, given that there was the option of having a colloquim in the laboratory instead of this project. Even so, the facial recognition and detection seemed way too interesting to avoid it.

5.2 Sliding window

Shortly explanined, the method of sliding window is a method of processing the images in which we go with a smaller image piece or with its descriptors (be it Hog, be it contrast, etc.) over a larger image and use the weights and bias of a classifier trained to obtain a score of the match/ correlation of the patch over the big image. To this score we impose a threshold, such that a similitude of the descriptors (be it Hog, contrast, colour, etc) will lead to a bigger score relative to the other image patches. Practically, by doing it so, one can obtain detections. Of course, the patch image does not have to be fixed, but have different sizes. As discussed in the laboratory, I have kept constant the small image but I varied the big image, giving it a 0.95 coefficient for reach run (the big image became 5% smaller with each run, until it reached a threshold such that it would not get smaller than the size of the patch).

5.3 The Hog cell size

In the project of facial detection, I have used Hog cells of 6 pixels and then of 9 and 3 pixels. The best results I obtained with 3 pixels, because the Hog descriptors could better follow the faces, the accuracy was better as the step was smaller. The 9 pixels case could not suggest face details as nose or eyes, but just the face contour. Even so, for the actual project, because the cartoons are way too expressive, and the training data are too different (oblical/horizontal figures, even the name of the TV station appeared in training images, etc) I observed that the small granularity is not really effective.

5.4 Hard negatives

Even starting from last years project I have seen the optional function of detecting hard negatives. I have implemented this function to eliminate some of the false positive detections. The method was as follows: I have fed to the model images in which there were no faces as test images, I have saved the descriptors that gave detections over a threshold score, false detections and I used just these descriptors again, as hard negatives. By this I refer to the set of descriptors that is the toughest, the ones that can show the most false detections with the easiest effort. Because of that I considered that I do not need the simple hard negatives descriptors, because these ones are the strongest negative descriptors. By doing so I not only reduced the .npy file in which I was holding the descriptors, but I also eliminated a lot of false positive detections. Practically, the score reached 75-80% and I decided to stop here and start the current project with cartoon figures, on which the homework is actually graded.

5.5 Code structure

As the input was images of 480*640 pixels with cartoons from the ”Life with Louie” TV series plus the annotations, I considered that it is useful to extract annotated faces from these images in separate folders, such that I know from which annotation and for whom I obtained one image. The difficulty here was to get right the annotation method, X and Y axis, the biggest coordinates being the top down-right, therefore this part seemed counter-intuitive to me and I had to run Python in Debug mode to be able to follow what happens with X and Y because everything is in fact mirrored. At this stage, in essence, I obtained the folder with the faces.

5.6 An important remark- data filtering

It is interesting that once I found Ora or Louie images in the folder in which I had to have only Andy images. Being many Andy images in that folder, I realized that the data should be cleaned such that I would not get erroneous detections. I implemented such a function called clean_data_heads() which deletes images with specific indexes that I supplied beforehand. Practically, all images shall always have same name and indexes, with each extraction and the delete of the problematic images is easy and can be repeated with each following deletion of the folders (for instance in the case in which one wishes to add some extra pixels to the heads of the characters).

5.7 Task 2- images from which I extract heads and bodies

My ideea from the start was that if I manage to detect each character (so to complete Task 2), then Task 1 would be the unification of the characters' detections, aditionally the Unknown characters (that are none of the four characters of interest). This means that in essence, I started with solving Task 2. This one needed images that did not contain the respective faces of the characters. Having the face cut from the images, I realized that easiest is to use the original image but without the face of the character of interest. Therefore I made the pixels from annotations 0, such that I replaced them with black (the technique was not the most elegant, as I extracted body in the same get_heads() function in different folders such that it would be easier for me to use them for training). It is interesting to note that moving from absolute black to absolute black has no gradient, Hog descriptors would not be influenced by these parts of the images, so a very favourable thing for the classification.

5.8 Necessity of saving the data

With each run, it became clear that I need not extract heads and the rest of the image (the so-called bodies) with each run, but it was necessary to save them to load them quickly for a new run. Therefore, I saved .npy-s and .jpg images and CSV-s, txt-s for later pentru processing. It is important to mention that I started to work on the folder structure presented in the PDF of the requirements for the project, as there were many paths that otherwise needed updates.

5.9 Dedicated Classifier for each character

Each character has its own features and a specific optimum threshold, therefore I formed several classifiers for each character, in addition to a vector of threshold values for each character. In the figure below it is shown the performance of a classifier for Louie.

5.10 Use of images for negatives Task 1

For Task 1, I did not think it was appropriate to use images without heads, because each was determined by taking out the information referring to the heads for each character, one by one. In other words, if for instance I would have used the image in which the head of Louie would not have appeared, certainly existed images with Louie with at least one of Andy, Ora, Tommy or Unknown. Thus, I got over to obtaining own images for this task, for the negatives. I did screen captures for different images in which I did not observe any character and I took care that these images would be 480*640 pixels as size. Then I used these images to obtain the negative Hog descriptors.

5.11 Detection of interesant part of an image- in the end, removed from the project but interesting for later improvements

Because the detections took so much time, I considered necessary to obtain rapidly the most interesting part of the image. I spent a night looking for informations when I saw the conclusions of [B. Alexe, T. Deselaers and V. Ferrari, "What is an object?," 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Francisco, CA, USA, 2010, pp. 73-80]. Starting from the idea of saliency from [X. Hou and L. Zhang. Saliency detection: A spectral residual approach. In CVPR, 2007], Professor B. Alexe, the instructor of this course, concludes with an improvement of approximatively 40x of the processing speed for the sliding window method [N. Dalal and B. Triggs. Histogram of Oriented Gradients for Human Detection. In CVPR, 2005]. Therefore, for the implementation of the detection of the salience, I have initially used the functions cv.saliency.StaticSaliencyFineGrained_create() and (success, saliencyMap) = saliency.computeSaliency(img) Presently, I left the implementation commented in the code in run(), because in the laboratory of the next day I found out that it is not permitted the use of tricks to decide which part of the image is more interesting, such that only that would be used to detect faces. Thus I gave up the code by commenting it, but the images obtained beforehand seemed interesting and I intend to use this transform after submitting the project for grading to obtain special images, especially with FineGrained() function. For these functions I installed an OpenCV library with more functions (opencv-contrib-python-4.7.0.68), I did not uninstall it because of this idea to get back to it at a later point in time, when time shall permit. In any case, in practice, in the night before the laboratory, in an image with Andy holding in hand papers (image 004.jpg from the validation set), the algorithm detected the papers as the most salient part of the image, not the face of Andy. Therefore it was even easier to give up this idea.

Below, several images relevant for this ideea that I gave up.

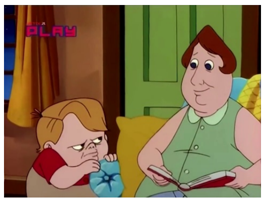


Figure 14: Original image 0002.jpg from the Validation set

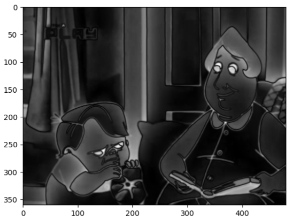


Figure 15: Image 0002.jpg with filter cv.saliency.StaticSaliencyFineGrained

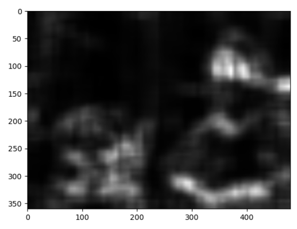


Figure 16: Image 0002.jpg with filter cv.saliency.StaticSaliencySpectralResidual

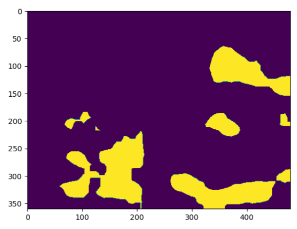


Figure 17: Imaginea 0002.jpg with filter cv.saliency.StaticSaliencySpectralResidual, threshold, dilation and erosion applied

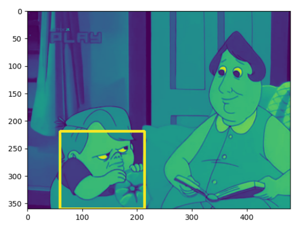


Figure 18: Image 0002.jpg after filter cv.saliency.StaticSaliencySpectralResidual, threshold, dilation and erosion and selection of components that give the largest area

Figure 19: Original image 0004.jpg from the Validation set

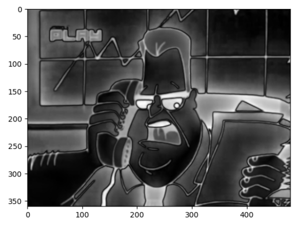


Figure 20: Image 0004.jpg with filter cv.saliency.StaticSaliencyFineGrained

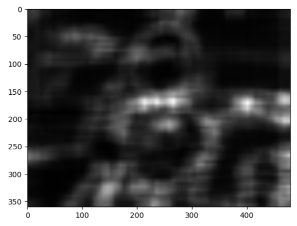


Figure 21: Image 0004.jpg with filter cv.saliency.StaticSaliencySpectralResidual

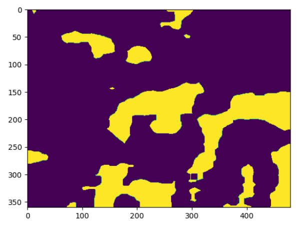


Figure 22: Image 0004.jpg with filter cv.saliency.StaticSaliencySpectralResidual, threshold, dilation and erosion

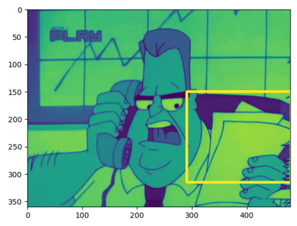
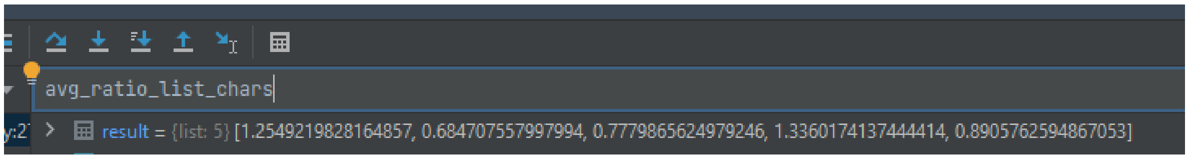


Figure 23: Image 0004.jpg after filter cv.saliency.StaticSaliencySpectralResidual, threshold, dilation and erosion and selection of the connected components of largest area

5.12 Aspect ratio for each character

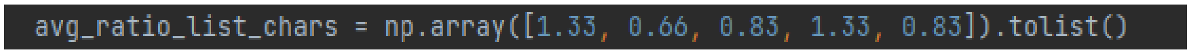
In the penultimate laboratory we were given a clue: different aspect ratio would come in very handy. I remembered I tried to see how would a non-squared window work, but obtained some errors and gave up the idea. Having a more advanced code in a way, I said to myself I should implement such a thing.

Computed ratio when it is used dinamically one gets errors because the Hog window changes dimensions. Therefore, I used computed ratio to adjust it in a convenient way in number of steps of 1/Cell_Hog. The dimension of the Hog cell used in the project is 6.



```
avg_ratio_list_chars
result = {list: 5} [1.2549219828164857, 0.684707557997994, 0.7779865624979246, 1.3360174137444414, 0.8905762594867053]
```

Figure 24: **Computed** ratio on the training data for Louie, Andy, Ora and Tommy- in this order



```
avg_ratio_list_chars = np.array([1.33, 0.66, 0.83, 1.33, 0.83]).tolist()
```

Figure 25: **Adjusted** ratio on the training data for Louie, Andy, Ora and Tommy- in this order

5.13 From ScikitLearn to OpenCV

Practically, the idea exposed above led me to giving up the function offered by ScikitLearn. I switched to another function that obtained Hog descriptors, but from OpenCV library. The reason is that it offered better control, but surprisingly, it proved that it was quicker for calculations. A search on StackOverflow showed that this function can give a speedup of 70x. I took the decision to try to modify the entire project and to go on this variant, I edited the implementation of the patch from the laboratory (instead of going from window to window on the Hog descriptor of the big window, I go on the big image piece by piece and I compute its Hog descriptor). Going over these obstacles, the calculation duration for each validation image has been well improved, with a factor of approximatively 20x. What was computed in an average of 60s, now it was necessary only 4s. For 200 validation images, the difference became 3.5h simulation time versus 12-15 minutes. The road has not been without difficulties, that aspect ratio could not be continosly adjusted, but approximated such that I could go over the image with cells of well-chosen dimensions.

5.14 Giving up hard negatives

Having the implementation with the Hog descriptors from the OpenCV library, and having a considerabl improvement, I observed that I could raise a lot the number of hard negatives and did not run into out-of-memory errors. Therefore, because testing test images without faces is like a validation on test images, each taking 4-10seconds and seeing that it is very rapid the calculation of the Hog descriptors now, even for 350.000 negative patches, I gave up computing hard negatives because of the time necessary for the calculation of these descriptors, a too large time. The performance seemed to be similar (a very large number of negative descriptors versus hard negatives).

5.15 Adding pixels to heads

It is interesting the idea exposed in the penultimate laboratory, that of taking a bit of extra information from the annotations. For Louie and Ora this technique is useful, but in the case of Andy, not necessarily. Andy had annotations in which the face was inclined or even rotated by 90 degrees, moreover, I observed in a lot of cases the badge of the TV station transmitting the cartoons, among the training pictures. For these reasons I think the expressiveness of this character led to a AP score on average smaller for it, compared to Louie, Ora and Tommy. Of course, I took out these extra pixels from the images that were left with the bodies after the head extractions.

6 Conclusions

6.1 Summary of difficulties met along the way

The hardest was for me to advance with the face detection, on the previous human face detection project. I changed different parameters until I got bold and implemented the sliding window implementation. Then, working with folders was a bit difficult, often the images were not read, I was missing some character that splits folders and did not realize that. Sometimes, I did not have detections and because of that I had write errors, I had to sort out this bug too.

The persistent difficulty I think was to be patient to let the simulations end, or to get to a point good enough with the code such that I would let simulations run until the morning.

6.2 What would I improve in the future

The code of the project shows still like a draft. I would have loved to have more time to clean it up from unnecessary comments, to organize it better, but these things are not being awarded points, however the AP metric yes, therefore I insisted on the performance obtained with the code, not the elegance of the code. Moreover, I would spend more time on Task 1, the task where I expected a better performance. For Task 2 (facial recognition) it seems by all chances that I could in principle obtain the bonus point on the test data, but for Task 1 (facial detection) I estimate I loose about 1 out of 4 points and that I do not get the extra point.

Because for the negative examples, the randomization is quite important, I would have wished I would use a randomization algorithm useful in criptography, there is a Python library with something like this. This would have shrank the percentage variation for a determined AP.

I would have made a function that would change the threshold for the score for descriptors such that it would keep the one that gives a maximum AP for each character. This would have saved me from running on 200 validation images for a new larger threshold. In essence, a larger threshold leads to a smaller AP, but to see the efficiency of eliminating false positive detections, it is interesting such a function.

I would have included a library that would contain a sound that is triggered when the simulation ended, such that I would not need to keep my eyes on the simulations.

I thought of trying AdaBoost instead of SVM. I would have tried the ideea described in [Sun, Weihan & Kise, Koichi. (2012). Cartoon Character Recognition Using Concentric Multi-Region Histograms of Oriented Gradients. IEEJ Transactions on Electronics, Information and Systems. 132. 1847-1854.], a HOG with higher granularity in the center of the face and a smaller one going towards the margins.

In my opinion, a promising solution would have been the neural networks for this task. But a first exploration of the idea on internet showed that it would require a longer time than SVM for processing, therefore I did not find the time necessary to explore this idea.

The project touched its purpose, that of applying the computer vision techniques presented in the course and laboratory for the face detection and recognition. The difference between the two is that in the first case, faces are being detected, in the second the character to which the face belongs is identified. I started with Task 2 (facial recognition) because I thought it is more interesting and I ended with Task 1 (facial detection). The result seems to be directly proportional with the effort for the tasks, chances of 1 point bonus being only for Task 2 for getting over the threshold of 60% mAP. At the time of writing this documentation, I did not manage to get over the threshold of 80% for Task 1.

6.3 General conclusions

To end, the two Computer Vision projects got my interest and got my very little free time. It is worth mentioning that the grade of this project is made of 4 points Task 1, 4 points Task 2, 1 point PDF documentation and one point from office only if there is a README file with indications of how to run the software and which libraries were used such that the effort made by the professors to run the code is minimal. One can get a bonus point on each Task if the threshold value of 80% is exceeded for AP (average precision) and of 60% mAP (mean average precision) for Task 2. The deadline to submit the code and the descriptors is very strict, **22 January 2023 23:59**.

Last but not least, I am very pleased that I got to use Python very well and that I am fully confident in the PyCharm 2022.2.3 (Professional Edition) software for debug and code implementation, step by step, in different projects (such as this one with quick visual feedback, the Computer Vision course).

In addition, I got used to writing in LaTeX, using it to write the documentation of the two projects, as soon in some months I am to defend the Bachelor of Science in Informatics Thesis using the same text processing language.

7 Annex

This annex shows folders with heads, with the images except the bodies and the code that I was forbidden from using in the project in order to get the salient areas of the pictures.

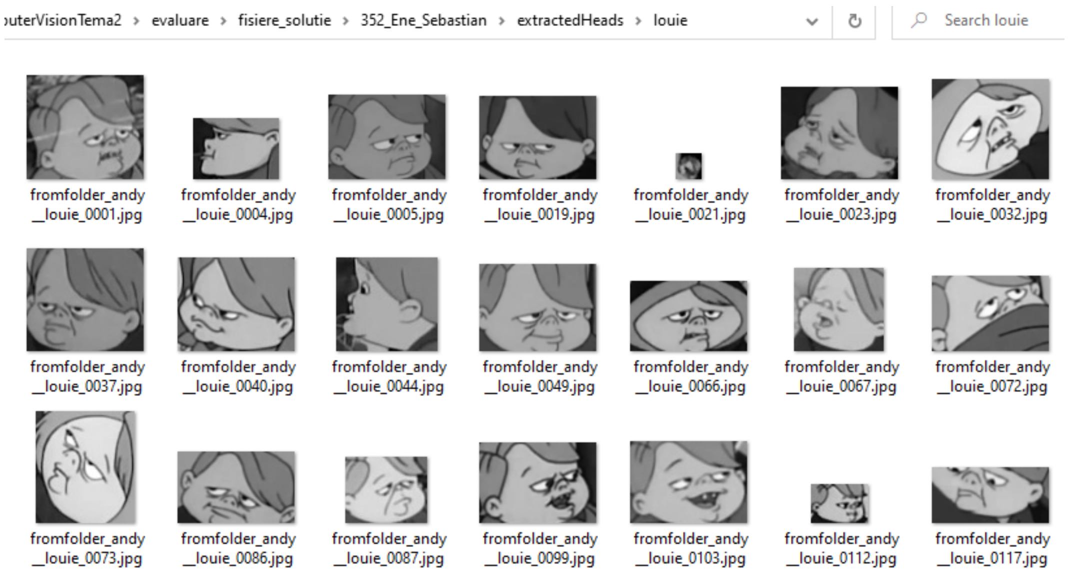

ɔuterVisionTema2 › evaluare › fisiere_solutie › 352_Ene_Sebastian › extractedHeads › louie
Search louie
fromfolder_andy__louie_0001.jpg
fromfolder_andy__louie_0004.jpg
fromfolder_andy__louie_0005.jpg
fromfolder_andy__louie_0019.jpg
fromfolder_andy__louie_0021.jpg
fromfolder_andy__louie_0023.jpg
fromfolder_andy__louie_0032.jpg
fromfolder_andy__louie_0037.jpg
fromfolder_andy__louie_0040.jpg
fromfolder_andy__louie_0044.jpg
fromfolder_andy__louie_0049.jpg
fromfolder_andy__louie_0066.jpg
fromfolder_andy__louie_0067.jpg
fromfolder_andy__louie_0072.jpg
fromfolder_andy__louie_0073.jpg
fromfolder_andy__louie_0086.jpg
fromfolder_andy__louie_0087.jpg
fromfolder_andy__louie_0099.jpg
fromfolder_andy__louie_0103.jpg
fromfolder_andy__louie_0112.jpg
fromfolder_andy__louie_0117.jpg

Figure 26: Example of content of the folder ExtractedHeads with Louie as images for the positive descriptors

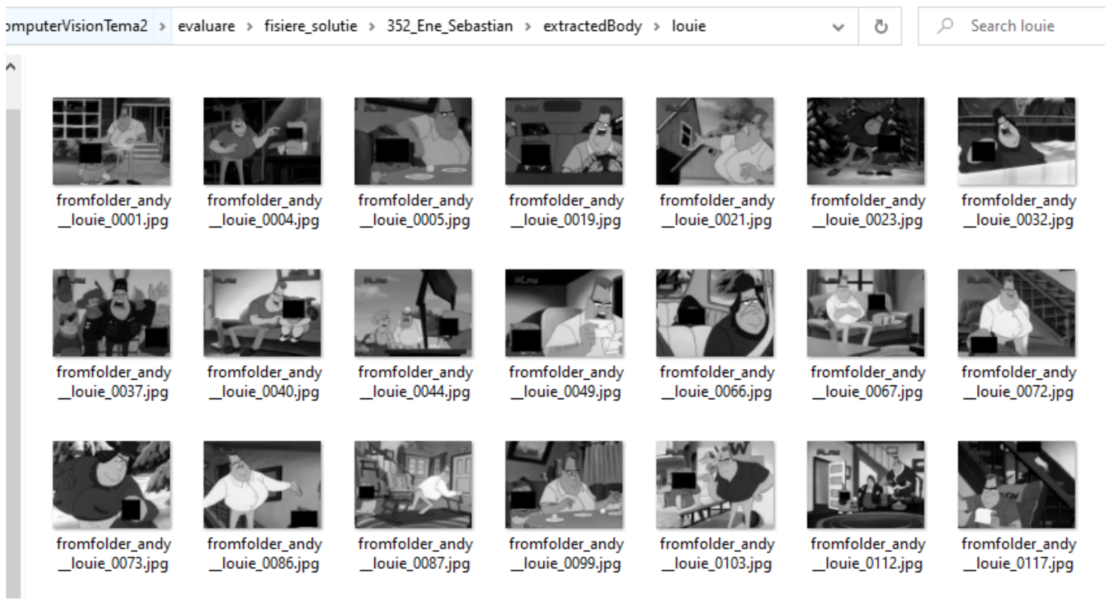

ɔmputerVisionTema2 › evaluare › fisiere_solutie › 352_Ene_Sebastian › extractedBody › louie
Search louie
fromfolder_andy__louie_0001.jpg
fromfolder_andy__louie_0004.jpg
fromfolder_andy__louie_0005.jpg
fromfolder_andy__louie_0019.jpg
fromfolder_andy__louie_0021.jpg
fromfolder_andy__louie_0023.jpg
fromfolder_andy__louie_0032.jpg
fromfolder_andy__louie_0037.jpg
fromfolder_andy__louie_0040.jpg
fromfolder_andy__louie_0044.jpg
fromfolder_andy__louie_0049.jpg
fromfolder_andy__louie_0066.jpg
fromfolder_andy__louie_0067.jpg
fromfolder_andy__louie_0072.jpg
fromfolder_andy__louie_0073.jpg
fromfolder_andy__louie_0086.jpg
fromfolder_andy__louie_0087.jpg
fromfolder_andy__louie_0099.jpg
fromfolder_andy__louie_0103.jpg
fromfolder_andy__louie_0112.jpg
fromfolder_andy__louie_0117.jpg

Figure 27: Example of content of the folder ExtractedBody with Louie as source images for sub-images as negative descriptors

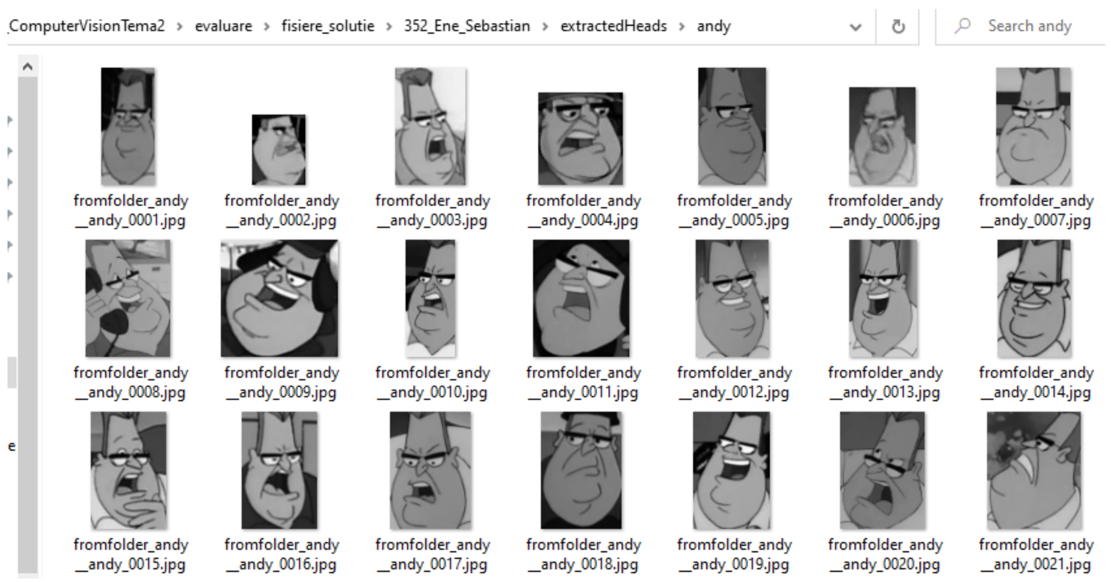

ComputerVisionTema2 › evaluare › fisiere_solutie › 352_Ene_Sebastian › extractedHeads › andy
Search andy
fromfolder_andy__andy_0001.jpg
fromfolder_andy__andy_0002.jpg
fromfolder_andy__andy_0003.jpg
fromfolder_andy__andy_0004.jpg
fromfolder_andy__andy_0005.jpg
fromfolder_andy__andy_0006.jpg
fromfolder_andy__andy_0007.jpg
fromfolder_andy__andy_0008.jpg
fromfolder_andy__andy_0009.jpg
fromfolder_andy__andy_0010.jpg
fromfolder_andy__andy_0011.jpg
fromfolder_andy__andy_0012.jpg
fromfolder_andy__andy_0013.jpg
fromfolder_andy__andy_0014.jpg
fromfolder_andy__andy_0015.jpg
fromfolder_andy__andy_0016.jpg
fromfolder_andy__andy_0017.jpg
fromfolder_andy__andy_0018.jpg
fromfolder_andy__andy_0019.jpg
fromfolder_andy__andy_0020.jpg
fromfolder_andy__andy_0021.jpg

Figure 28: Example of content of the folder ExtractedHeads with Andy as images for the positive descriptors

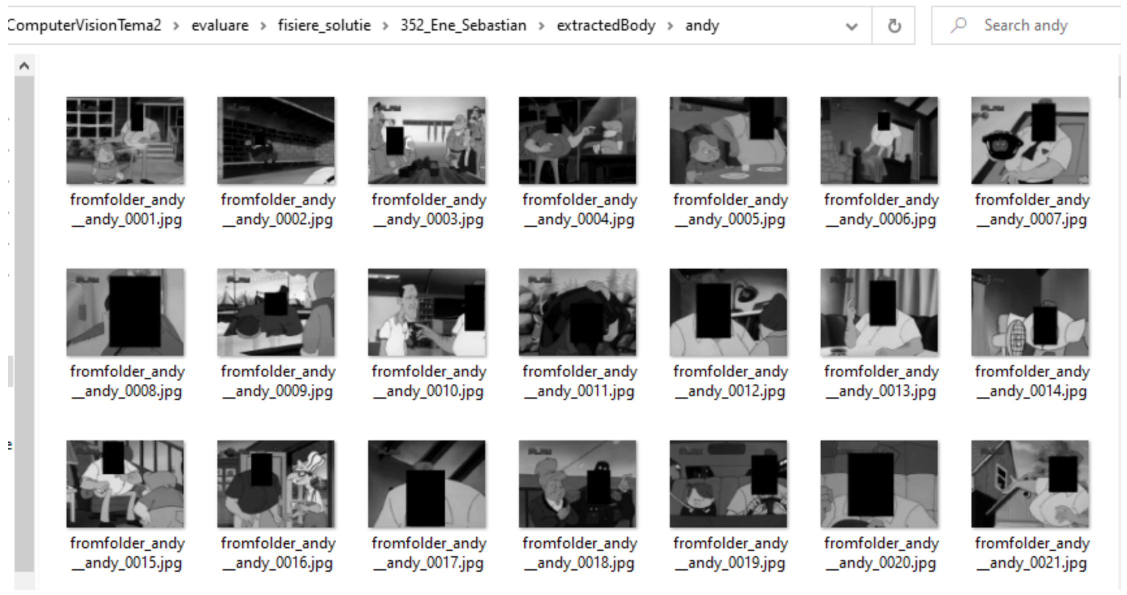

ComputerVisionTema2 › evaluare › fisiere_solutie › 352_Ene_Sebastian › extractedBody › andy
Search andy
fromfolder_andy__andy_0001.jpg
fromfolder_andy__andy_0002.jpg
fromfolder_andy__andy_0003.jpg
fromfolder_andy__andy_0004.jpg
fromfolder_andy__andy_0005.jpg
fromfolder_andy__andy_0006.jpg
fromfolder_andy__andy_0007.jpg
fromfolder_andy__andy_0008.jpg
fromfolder_andy__andy_0009.jpg
fromfolder_andy__andy_0010.jpg
fromfolder_andy__andy_0011.jpg
fromfolder_andy__andy_0012.jpg
fromfolder_andy__andy_0013.jpg
fromfolder_andy__andy_0014.jpg
fromfolder_andy__andy_0015.jpg
fromfolder_andy__andy_0016.jpg
fromfolder_andy__andy_0017.jpg
fromfolder_andy__andy_0018.jpg
fromfolder_andy__andy_0019.jpg
fromfolder_andy__andy_0020.jpg
fromfolder_andy__andy_0021.jpg

Figure 29: Example of content of the folder ExtractedBody with Andy as source images for sub-images as negative descriptors

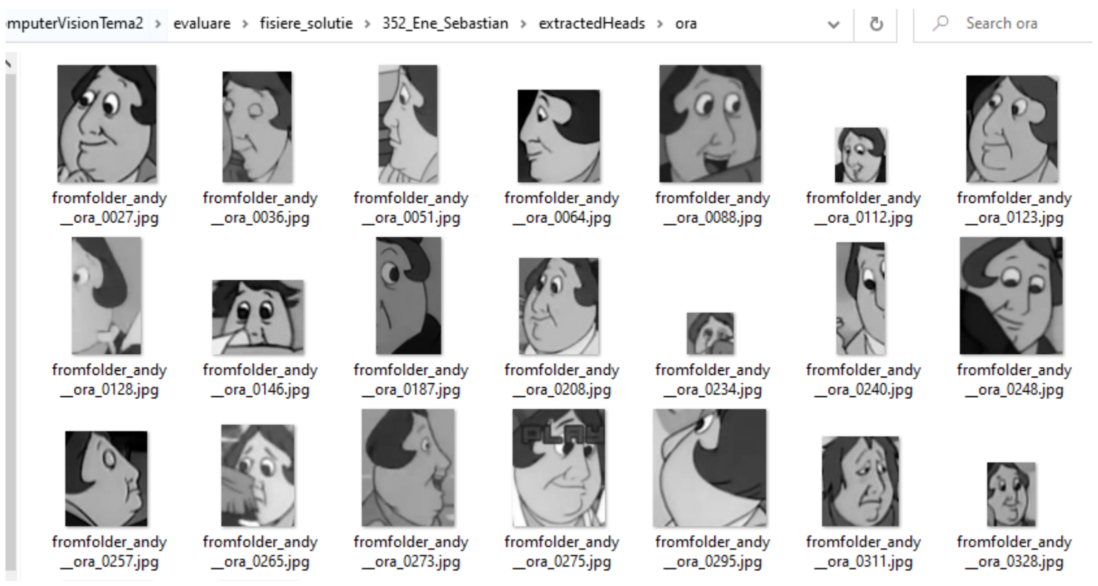

mputerVisionTema2 › evaluare › fisiere_solutie › 352_Ene_Sebastian › extractedHeads › ora
Search ora
fromfolder_andy__ora_0027.jpg
fromfolder_andy__ora_0036.jpg
fromfolder_andy__ora_0051.jpg
fromfolder_andy__ora_0064.jpg
fromfolder_andy__ora_0088.jpg
fromfolder_andy__ora_0112.jpg
fromfolder_andy__ora_0123.jpg
fromfolder_andy__ora_0128.jpg
fromfolder_andy__ora_0146.jpg
fromfolder_andy__ora_0187.jpg
fromfolder_andy__ora_0208.jpg
fromfolder_andy__ora_0234.jpg
fromfolder_andy__ora_0240.jpg
fromfolder_andy__ora_0248.jpg
fromfolder_andy__ora_0257.jpg
fromfolder_andy__ora_0265.jpg
fromfolder_andy__ora_0273.jpg
fromfolder_andy__ora_0275.jpg
fromfolder_andy__ora_0295.jpg
fromfolder_andy__ora_0311.jpg
fromfolder_andy__ora_0328.jpg

Figure 30: Example of content of the folder ExtractedHeads with Ora as images for the positive descriptors

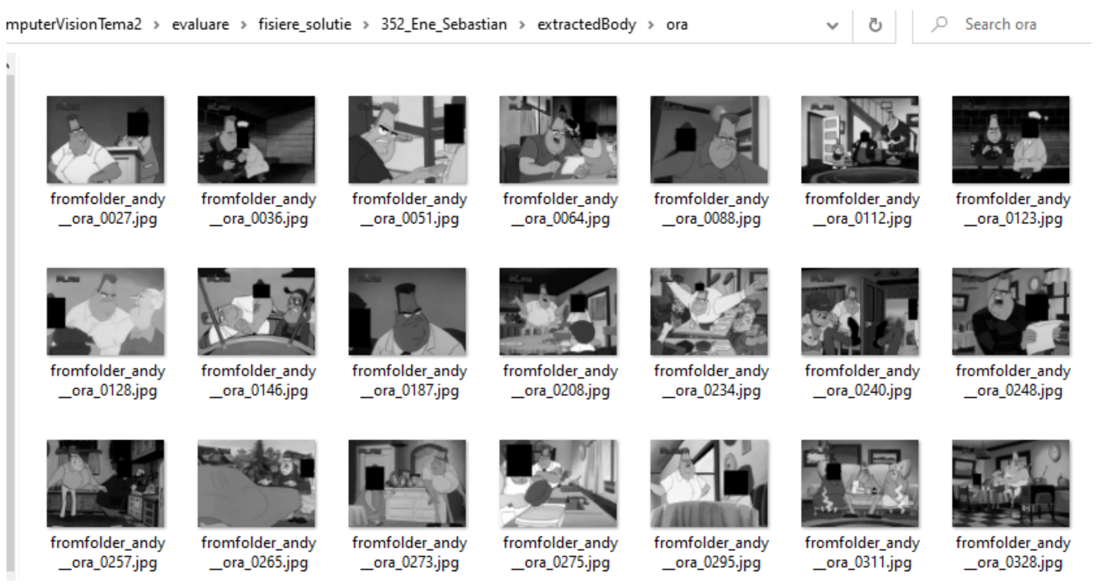

mputerVisionTema2 › evaluare › fisiere_solutie › 352_Ene_Sebastian › extractedBody › ora
Search ora
fromfolder_andy__ora_0027.jpg
fromfolder_andy__ora_0036.jpg
fromfolder_andy__ora_0051.jpg
fromfolder_andy__ora_0064.jpg
fromfolder_andy__ora_0088.jpg
fromfolder_andy__ora_0112.jpg
fromfolder_andy__ora_0123.jpg
fromfolder_andy__ora_0128.jpg
fromfolder_andy__ora_0146.jpg
fromfolder_andy__ora_0187.jpg
fromfolder_andy__ora_0208.jpg
fromfolder_andy__ora_0234.jpg
fromfolder_andy__ora_0240.jpg
fromfolder_andy__ora_0248.jpg
fromfolder_andy__ora_0257.jpg
fromfolder_andy__ora_0265.jpg
fromfolder_andy__ora_0273.jpg
fromfolder_andy__ora_0275.jpg
fromfolder_andy__ora_0295.jpg
fromfolder_andy__ora_0311.jpg
fromfolder_andy__ora_0328.jpg

Figure 31: Example of content of the folder ExtractedBody with Ora as source images for sub-images as negative descriptors

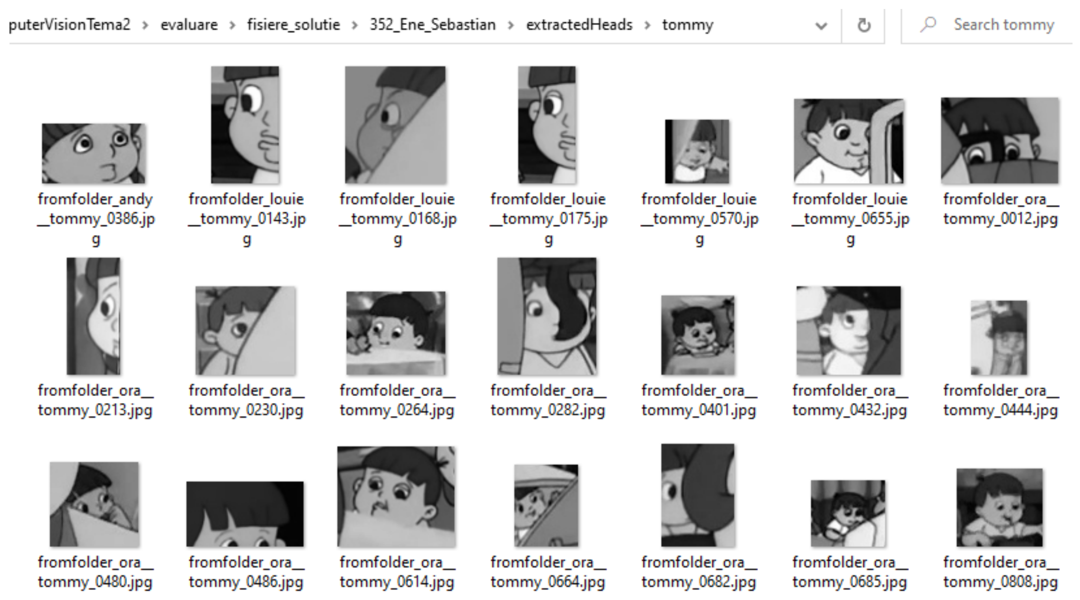

puterVisionTema2 › evaluare › fisiere_solutie › 352_Ene_Sebastian › extractedHeads › tommy
Search tommy
fromfolder_andy__tommy_0386.jpg
fromfolder_louie__tommy_0143.jpg
fromfolder_louie__tommy_0168.jpg
fromfolder_louie__tommy_0175.jpg
fromfolder_louie__tommy_0570.jpg
fromfolder_louie__tommy_0655.jpg
fromfolder_ora__tommy_0012.jpg
fromfolder_ora__tommy_0213.jpg
fromfolder_ora__tommy_0230.jpg
fromfolder_ora__tommy_0264.jpg
fromfolder_ora__tommy_0282.jpg
fromfolder_ora__tommy_0401.jpg
fromfolder_ora__tommy_0432.jpg
fromfolder_ora__tommy_0444.jpg
fromfolder_ora__tommy_0480.jpg
fromfolder_ora__tommy_0486.jpg
fromfolder_ora__tommy_0614.jpg
fromfolder_ora__tommy_0664.jpg
fromfolder_ora__tommy_0682.jpg
fromfolder_ora__tommy_0685.jpg
fromfolder_ora__tommy_0808.jpg

Figure 32: Example of content of the folder ExtractedHeads with Tommy as images for the positive descriptors

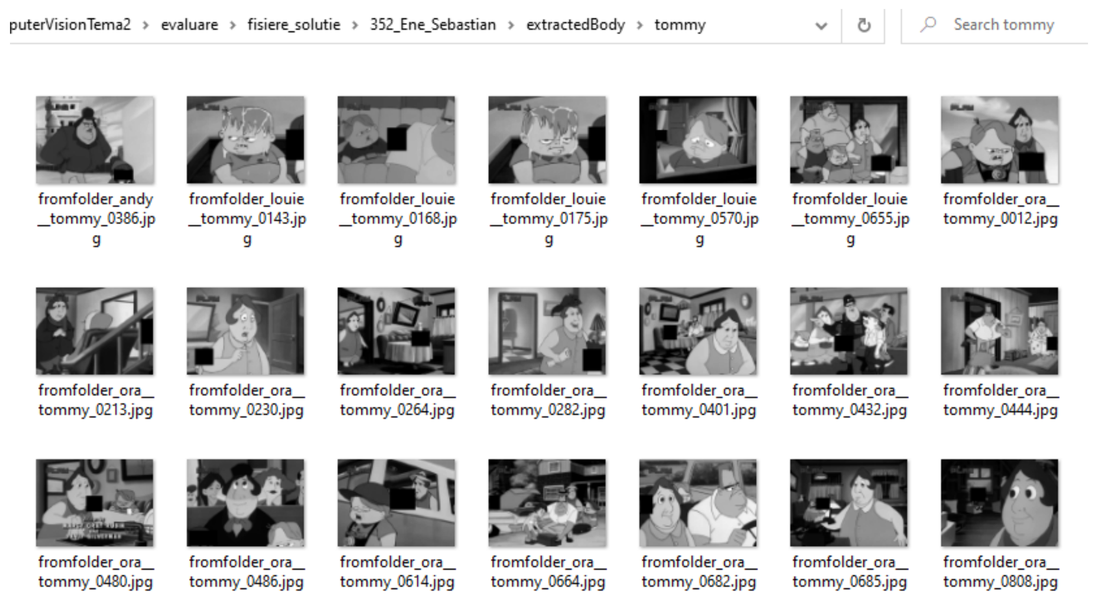

puterVisionTema2 › evaluare › fisiere_solutie › 352_Ene_Sebastian › extractedBody › tommy
Search tommy
fromfolder_andy__tommy_0386.jpg
fromfolder_louie__tommy_0143.jpg
fromfolder_louie__tommy_0168.jpg
fromfolder_louie__tommy_0175.jpg
fromfolder_louie__tommy_0570.jpg
fromfolder_louie__tommy_0655.jpg
fromfolder_ora__tommy_0012.jpg
fromfolder_ora__tommy_0213.jpg
fromfolder_ora__tommy_0230.jpg
fromfolder_ora__tommy_0264.jpg
fromfolder_ora__tommy_0282.jpg
fromfolder_ora__tommy_0401.jpg
fromfolder_ora__tommy_0432.jpg
fromfolder_ora__tommy_0444.jpg
fromfolder_ora__tommy_0480.jpg
fromfolder_ora__tommy_0486.jpg
fromfolder_ora__tommy_0614.jpg
fromfolder_ora__tommy_0664.jpg
fromfolder_ora__tommy_0682.jpg
fromfolder_ora__tommy_0685.jpg
fromfolder_ora__tommy_0808.jpg

Figure 33: Example of content of the folder ExtractedBody with Tommy as source images for sub-images as negative descriptors

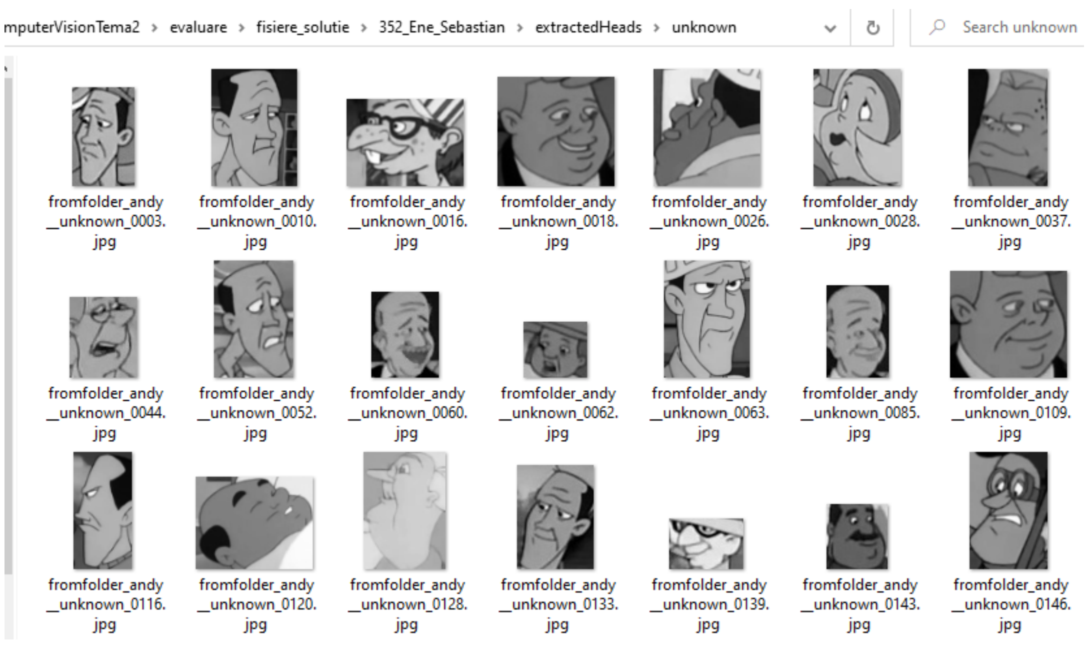

mputerVisionTema2 > evaluare > fisiere_solutie > 352_Ene_Sebastian > extractedHeads > unknown
Search unknown
fromfolder_andy__unknown_0003.jpg
fromfolder_andy__unknown_0010.jpg
fromfolder_andy__unknown_0016.jpg
fromfolder_andy__unknown_0018.jpg
fromfolder_andy__unknown_0026.jpg
fromfolder_andy__unknown_0028.jpg
fromfolder_andy__unknown_0037.jpg
fromfolder_andy__unknown_0044.jpg
fromfolder_andy__unknown_0052.jpg
fromfolder_andy__unknown_0060.jpg
fromfolder_andy__unknown_0062.jpg
fromfolder_andy__unknown_0063.jpg
fromfolder_andy__unknown_0085.jpg
fromfolder_andy__unknown_0109.jpg
fromfolder_andy__unknown_0116.jpg
fromfolder_andy__unknown_0120.jpg
fromfolder_andy__unknown_0128.jpg
fromfolder_andy__unknown_0133.jpg
fromfolder_andy__unknown_0139.jpg
fromfolder_andy__unknown_0143.jpg
fromfolder_andy__unknown_0146.jpg

Figure 34: Example of content of the folder ExtractedHeads with Unknown as images for the positive descriptors

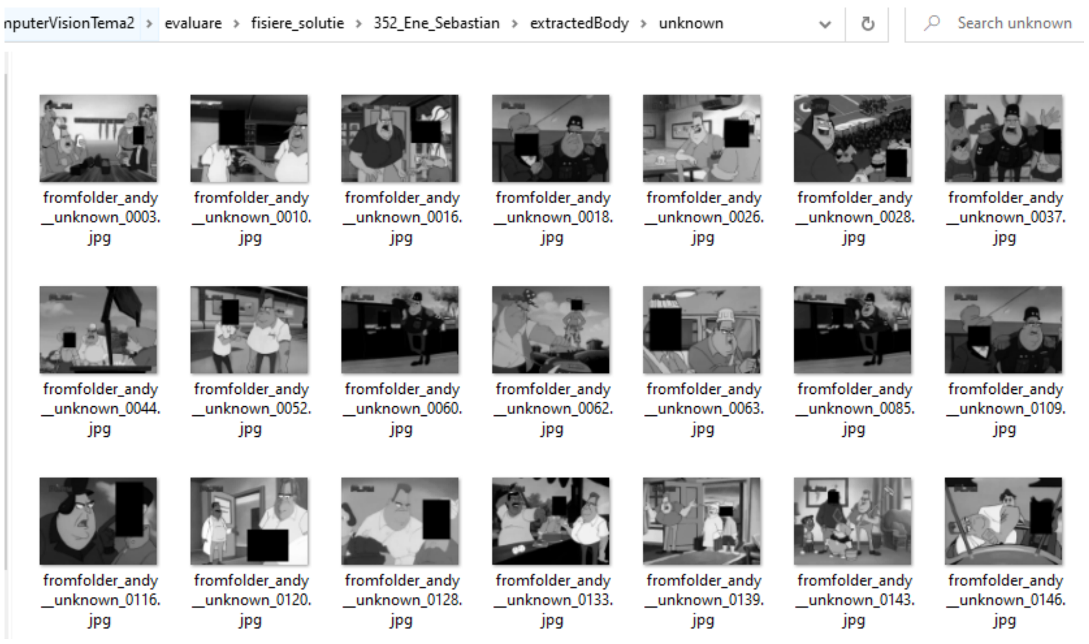

nputerVisionTema2 > evaluare > fisiere_solutie > 352_Ene_Sebastian > extractedBody > unknown
Search unknown
fromfolder_andy__unknown_0003.jpg
fromfolder_andy__unknown_0010.jpg
fromfolder_andy__unknown_0016.jpg
fromfolder_andy__unknown_0018.jpg
fromfolder_andy__unknown_0026.jpg
fromfolder_andy__unknown_0028.jpg
fromfolder_andy__unknown_0037.jpg
fromfolder_andy__unknown_0044.jpg
fromfolder_andy__unknown_0052.jpg
fromfolder_andy__unknown_0060.jpg
fromfolder_andy__unknown_0062.jpg
fromfolder_andy__unknown_0063.jpg
fromfolder_andy__unknown_0085.jpg
fromfolder_andy__unknown_0109.jpg
fromfolder_andy__unknown_0116.jpg
fromfolder_andy__unknown_0120.jpg
fromfolder_andy__unknown_0128.jpg
fromfolder_andy__unknown_0133.jpg
fromfolder_andy__unknown_0139.jpg
fromfolder_andy__unknown_0143.jpg
fromfolder_andy__unknown_0146.jpg

Figure 35: Example of content of the folder ExtractedBody with Unknown as source images for sub-images as negative descriptors

Extract of the Python code for saliency detection, principle implementation of the FFT spectral residuum (Fast Fourier Transform) [X. Hou and L. Zhang. Saliency detection: A spectral residual approach. In CVPR, 2007], it is not part of the present implementation, but for future implementations:

```
 #### Parte de extragere a partii interesante din imagini
# saliency = cv.saliency.StaticSaliencyFineGrained_create()
saliency = cv.saliency.StaticSaliencySpectralResidual_create()
(success, saliencyMap) = saliency.computeSaliency(img)
saliencyMap= (saliencyMap/np.max(saliencyMap) * 255).astype("uint8")
plt.imshow(saliencyMap, cmap="gray")
plt.show()

threshMap = cv.threshold(saliencyMap, 0, 255,
                         cv.THRESH_BINARY + cv.THRESH_OTSU)[1]
kernel = np.ones((10, 10), np.uint8)
threshMap = cv.dilate(threshMap, kernel, iterations=1)
threshMap = cv.erode(threshMap, kernel, iterations=1)
plt.imshow(threshMap)
plt.show()

contours, hierarchy = cv.findContours(threshMap * 1, cv.RETR_LIST,
cv.CHAIN_APPROX_SIMPLE)
contours = sorted(contours, key=cv.contourArea)

x, y, width, height  = cv.boundingRect(contours[-1])
cv.rectangle(img, (x, y), (x + width, y + height), (255, 0, 0), 3)
#img= cv.cvtColor(img, cv.COLOR_GRAY2RGB)
plt.imshow(img)
plt.show()
# bgdmodel = np.zeros((1, 65), np.uint8)
# fgdmodel = np.zeros((1, 65), np.uint8)
# saliencyMap= saliencyMap//130
# saliencyMap[np.where(saliencyMap >= 130)] = cv.GC_FGD
mask = saliencyMap
#img= cv.cvtColor(img,cv.COLOR_GRAY2RGB)
#cv.grabCut(img, mask, rect, bgdmodel, fgdmodel, 1, cv.GC_INIT_WITH_RECT)
# mask = np.where((mask == 2) | (mask == 0), 0, 1).astype('uint8')
plt.imshow(mask*img, cmap="gray")
plt.show()

plt.imshow(threshMap, cmap="gray")
plt.show()
plt.imshow(img, cmap="gray")
plt.show()
plt.imshow(saliencyMap, cmap="gray")
plt.show()
```