Fundamentals of image processing - Project

Andrea Giuseppe Zarola, Lorenzo Sasso, Mattia Ventola

February 12, 2021

Introduction

In our project we implemented an image annotation tool to allow users to crop objects from images. All the objects cropped by the user are saved in a specific output folder in order to be usable by machine learning models. In the next sections we reported the libraries that we used, a diagram to explain the workflow of the application and also the main steps that we implemented together with a pseudo-code. Finally we reported a little evaluation in which we explain some advantages and disadvantages of the final application.

Libraries

In order to work on the project we used some Python packages.

- **Tkinter**: it's a very useful Python package that allows to create in an easy way graphical user interfaces on Python. We used it to create the GUI of our project.
- **OpenCV**: it's an open source computer vision and machine learning software library that can be easily used inside Python. We needed to use it in our project in order to crop objects from images.
- **Pillow**: it's a Python imaging library that we used in our project in order to display the images processed with OpenCV inside the GUI.

Flow chart

Figure 1 represents the flow chart of all the operations that a user can do when uses our tool.

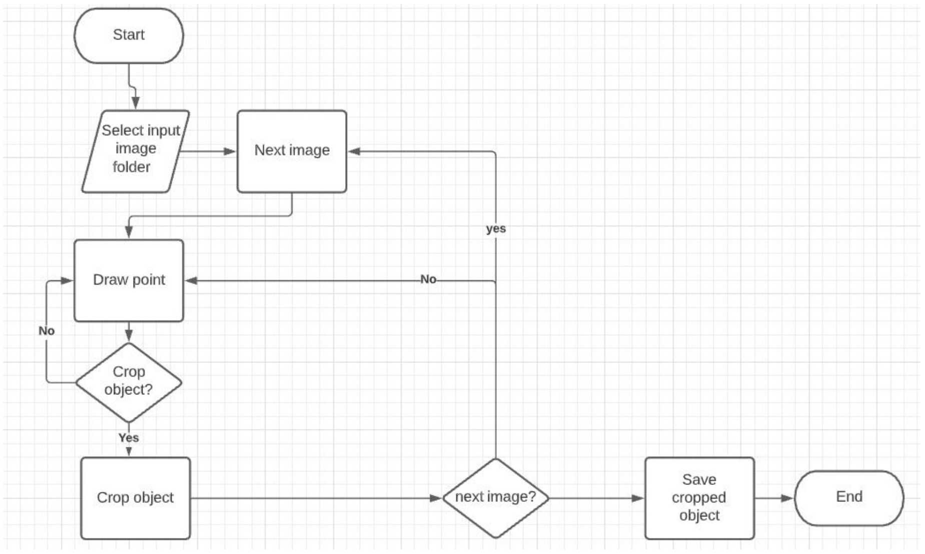


Figure 1: How our application works

Algorithm steps

We can report the major steps that are doing in our application with particular attention to the `crop_object` function that we defined to crop the object selected by the user.

1. First we create the GUI of our application that initially is only composed by four buttons.

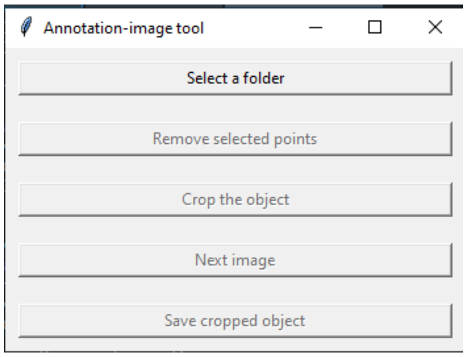
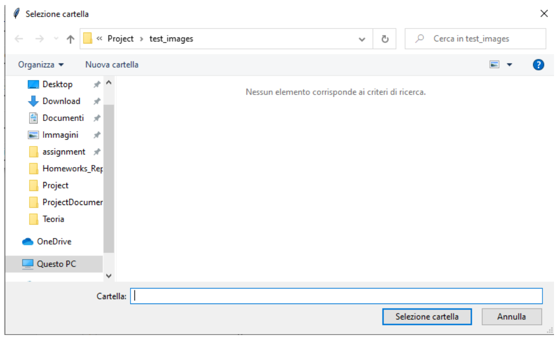


Figure 2: Step 1

2. The next step regards the selection of the folder by the user. Doing this, then we read using the `io` Python package all the images in that folder.



Nessun elemento corrisponde ai criteri di ricerca.

Figure 3: Step 2

3. After all this, the application present to the user the first image of the folder on which the user can select the boundaries of the object that he want to crop.

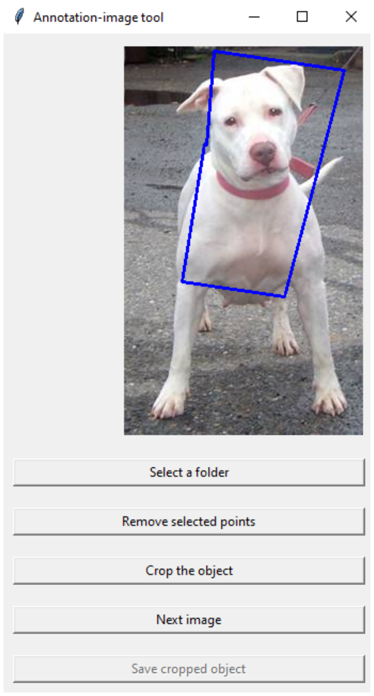


Figure 4: Step 3

4. When the user has chosen all the boundaries and clicked on the "Crop the object" button, we show in a different panel the result of the cropping operation.

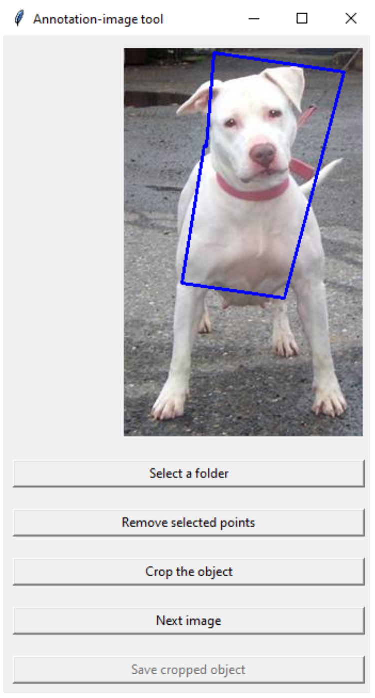


Figure 5: Step 3

5. Then the user can select "Next image" if want to continue to crop other object of the next images.

6. When the user want to save his work or has finished with all the images, it can save all the cropped objects by clicking on the "Save cropped objects" button.

Now we focus on the `crop_object` function that crops an object every time the user has already drawn some points (stored like a vector of coordinates in the image) on the image and has then has clicked on the "Crop the object" button.

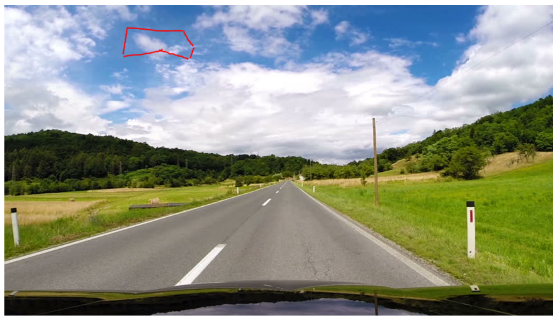


Figure 6: Cropping of the selected object.

1. First, we extract the rectangle which contains the cropped object and we extract the top-left coordinate and the width and height of the rectangle.
2. Then we crop the original image depending on the value of the top-left coordinate, the width and the height.

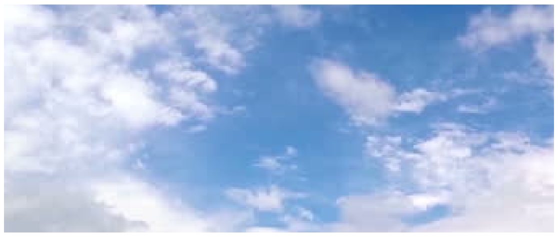


Figure 7: Step 2

3. Next, we create a mask with the same size of the rectangle cropped in the previous point and we draw the shape of the cropped image.

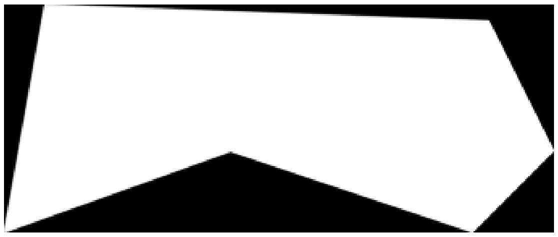


Figure 8: Step 3

4. Then, we calculate the per-element bitwise and conjunction between the mask and the cropped image.

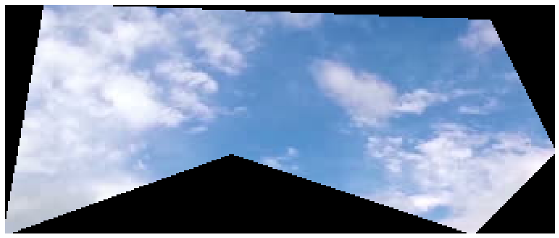


Figure 9: Step 4

5. Then, we add the white background through a mask that is the result of a bitwise not conjunction between white image and the mask in the point 2.

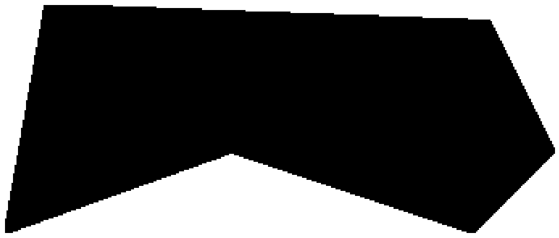


Figure 10: Step 5

6. Finally, we create the final cropped object that is the result of the sum among the mask in the point 5 and the image in point 4

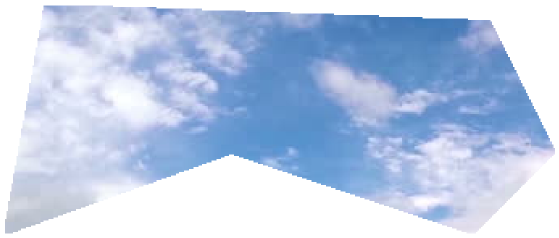


Figure 11: Step 6

Pseudo-code

We reported the pseudo-code that is implemented in the `crop_object` function.

```
points := input points
rect := top-left coordinate and the width and height
    of the rectangle which contains the input points
```

```
cropped := orginalImage[rect-coordinates]
mask := drawContours(mask, [pts]) # create a black
    mask with shape of cropped image in white
dst := bitwise_and(cropped, cropped, mask)
bg := bitwise_not(bg,bg, mask=mask) #create white
    background and black shape of the cropped image
croppedObject := bg + dst
```

Evaluation

At the end of our implementation, we can evaluate some aspects of our tool by viewing some useful controls that we implemented but also some improvements that can be made.
Regarding the GUI, it seems to be easily usable since at the start the user is presented with four different buttons, but only one of them ("Select a folder") is active. So, this let the user to easily understand that the first operation to do with the tool is to select a proper folder. Moreover, if the user choose to select a folder but then it doesn't select really the folder but instead clicks on the "Cancel" button then the only option for the user remains always the same. In general, every time the user knows what are the operations that he can do since we control this aspect by enable/disable the clickable buttons dynamically.
However, regarding the GUI, there are also some useful aspects that were not implemented. For instance, a good addition may be to allow the user to add some new images when he finished with the all the images contained in the folder selected at the start. Another interesting aspects may be to allow the user to return to a previous image on the folder, since we implemented only the "Next image" button. In addition, it's possible to think also some features to let the user to personalize the tool.
Regarding the "Crop object" operation, we implemented an important aspect that must be taken into account, that is to enable the cropping operation only when at least 3 points are selected. Moreover to allow the user to better understand what will be the cropped object, every time the tool draws some lines between the selected points in order to show to the user the selected region on the image.
Although when we tested the application we had not any problem to crop objects of any shape, some improvements can be made. For instance, we can think to add much more flexibility to the boundaries selection by adding the possibility to mark rounded lines for the boundaries. Although this may seem a serious shortcoming, in our application there's not limit to the maximum number of points that the user can mark to crop an object: for this reason, even if the user has to do a bit more work on tool it can crop also objects with rounded boundaries.