Dr. Asieh Parsania
Institut für Mathematik
Universität Zürich

MAT101: Programming

Group Project: Towards image privacy

Maria Han Veiga

Abstract

The proposed project is motivated by the increase in the usage of automated image recognition algorithms, and the paranoia induced by this. While the end goal might be quite ambitious, the project is aimed mainly to develop comfort coding in python, the initial contact with image processing techniques and familiarity with the **opencv** package. The project starts with developing a program to load images, and to blur and unblur. Furthermore, a face recognition algorithm should be developed (by, for example, using simple techniques available in **opencv** package). The first minimal viable product is comprised of a program that can load and display an image, a face detection algorithm and a blur/unblur filter.

Upon being successful in this task, more image processing techniques should be added to the programme, namely, image transformations that minimize the change in the input image perceptually, but that degrade the performance of the face detection algorithm. At the same time, better face detection algorithms can be implemented/tested.

1 Introduction

Imagine a dystopian future (which is not currently happening at all) where governments use facial recognition for racial profiling, tracking and control of minorities. And that this technology is used also to spot and fine jaywalkers, verify students at school gates, and monitor their expressions in lessons to ensure they are paying attention [1]. And that worse, many (many) governments start employing similar technologies. Wouldn't that be creepy?

This project is aimed at understanding face detection technology, and how to beat it. In figure 1, a simple face detection (Haar cascade [2]) algorithm is employed.

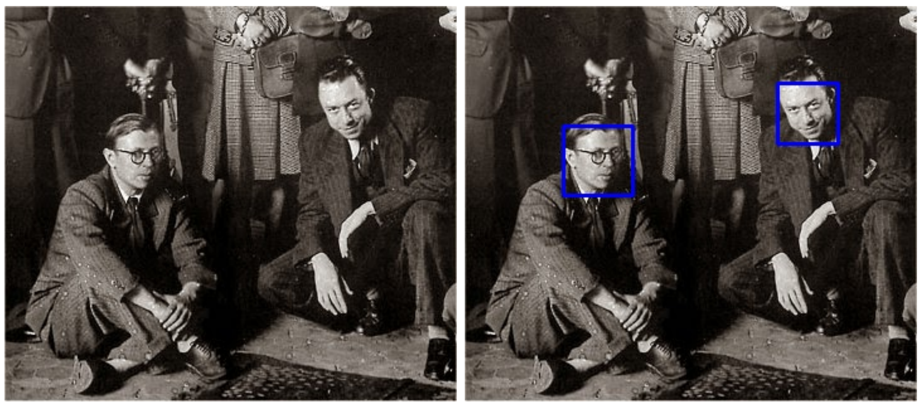

Figure 1: Simple face detection algorithm.

In figure 2, the picture is modified in such a way that the face detection algorithm does not find any faces anymore. The problem is, the initial picture has changed a lot, to the point that the people in the picture are no longer recognisable! So, there must be a middle ground between modifying the picture in a way that not too much information is lost, but enough information is lost that faces cannot be detected anymore. This is what you should work on in this project.

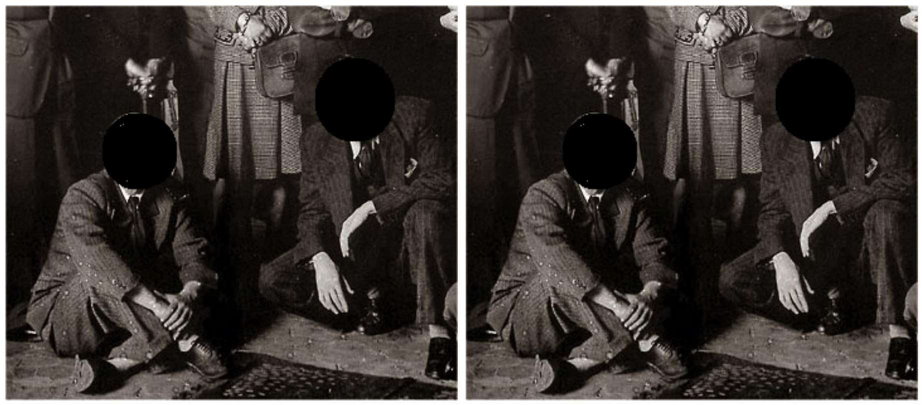

Figure 2: Simple face detection algorithm on a "privacy preserving picture". (No faces are detected).

2 Prerequisites / Information

This project will use python, scipy, numpy, matplotlib and opencv. You should know the basics from the classes and everything we have taught in the course so far. You should be willing to learn something about image processing. Furthermore, you should be willing to learn about good coding practices - virtual environments, code versioning and awareness of PEP 8 coding standards. A GUI library or other packages can be used to make a GUI. A browser based GUI is also acceptable.

3 Guidelines

The first set of tasks (0 to 2) should be completed in group, as it defines the setup of your project. After that, tasks 3 to 7, can be developed in parallel. Task 8 should be kept in mind, as it is a way to track your overall progress during these weeks.

You are encouraged to come to the first class after the midterm (Friday), where I will introduce code versioning and virtual environments in python. Afterwards, the Friday class will be for you to ask questions and to discuss conceptual problems / implementation problems. You can use all resources available on the web, but make sure you are not plagiarizing people's work. I will read your code!

Task 0
Preliminaries.

(a) Create a git-hub repository where your code should be hosted

(b) Set-up a virtual environment for the project

(c) Make sure everyone has access to the git-hub repository, can push and pull updates and has the right virtual environment.

(d) Be aware of PEP 8 coding standards (`https://www.python.org/dev/peps/pep-0008/`)

Task 1
Basics. This first step should be done in group. It defines the structure of the code that you will then extend.

(a) Write a function that can load and display an image

(b) Write a class which contains the data of an image

(c) Write a function which contains the face detection algorithm

(d) Write a function which can blur and unblur the loaded image

(e) Think of a nice way to structure your code. Try to answer questions such as: can I easily change the image? Where do my functions fail? If I want to add new features (e.g. different type of blurring), is it easy to do so? Is my python file super long (it shouldn't be!)? If I give the code to someone, will they understand what I am doing?

Task 2
Validation. Before you split up into individual tasks, you should make sure the code works. To validate the code, try the following things:

(a) Blur an image and unblur the image to see if the output is sensible. Do faces still get detected when blurred?

(b) Try to use different images (different sizes, black and white, with faces or without).

(c) You probably realised that the face detection does not work very well - sometimes faces are detected on things which are not faces, sometimes faces are not detected.

Tasks that can be done in parallel:

Task 3
Improving face detection algorithm. There are many ways to improve face detection algorithms. You want a set of functions which accepts as input an image (at least), and outputs an image with the faces detected. Some examples of things you can check out are:

- Local Binary Pattern
- Improving Haar Cascade with eye detection: `https://cv-tricks.com/computer-vision/case-study-training-better-haar-based-object-detectors/`

Task 4
Measuring similarity between pictures. Implement a metric that measures the similarity between two pictures. This is important, when you modify a picture in order to *confuse* the face detection algorithm, you probably want to have a way to quantify how much you have modified the original picture. There's many ways to do this. For example, you could just take the squared difference of the intensity of pixels, pixel by pixel. But for example, this metric might give you a large 'difference' when the perceived image (by human eye) does not look so different. You will want to have a (set of) function(s) that measure the similarity of two pictures and outputs some sort of metric.

Task 5
Selective application of filters (e.g selective blurring) Instead of blurring the whole picture, you might want to select parts of the image to blur, e.g. to not degrade the overall information of your picture. You will want to have a (set of) function(s) which allow you to specify a location in the image to apply the filters only in the given locations.

Task 6
Adding at least one algorithm that leads to better blurring/deblurring techniques or different filters (e.g. edge detection, adding noise, etc)

Task 7
Add a GUI that allows the user to load an image, apply filters and save the output. So far, the described code has been run through the terminal or your IDE of choice. In order to make it more user friendly, you might want to have a GUI that allows the user to load images, choose filters, save the output, test the face detection algorithm. You can use web-GUIs (they are faster!).

Before you run out of time, make sure you tested your contributions and that you have integrated your work with your colleagues work. Make sure you keep the git-hub up do date and...

Task 8
Try to fool the face detection algorithm! And make sure you save some examples for your presentation!

Bonus Tasks: Bonus tasks are open ended, you may do anything here. If you are not sure, please feel free to ask.

(a) Replace the face detection with object detection / recognition. Now, your programme is a lot more general!

(b) Realtime face detection

(c) Face recognition algorithm (e.g. using eigenfaces)

(d) Think of a bonus task on your own, ask me to confirm, be creative!

4 General Notes

- The goal of this project is to experience programming in a group. Discuss the project as a group and then divide the tasks among yourselves.
- You should discuss your progress with the supervisor of the project. Whenever you have questions about your project, feel free to ask them during the exercise class or post them in the forum.
- Once you have written your code you should briefly describe your results. You should include interesting examples and illustrations (if they are part of your project).
- To hand in your project, just send an email to the supervisor of your project. Make sure that it is clear who is responsible for which task.
- It is important that you understand the entire code of your group, not just the part that you have written yourself. In particular, you should be familiar with the prerequisites.
- During the last week of the semester, every group will have a 20-minute presentation of their project. Each member of the group should prepare a 4-minute presentation of their own code and be ready to answer questions about the entire project.
- For the project you will be graded as a group, but for the presentation you will be graded individually. Together the project and the presentation account for 30 percent of your final grade.
- The presentations will take place during the last week of the semester.

References

[1] U. of Technology Sydney, "Face recognition technology." https://www.uts.edu.au/about/faculty-law/news/face-recognition-technology. Accessed on 2019-10-20.

[2] OpenCV, "Cascade classifier." https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html. Accessed on 2019-10-20.