Institute for Visualization and Interactive Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Web Interface for Interactive Face Reconstruction

Lewin Fritzenschaft

Course of Study:	Softwaretechnik
Examiner:	Prof. Dr. Andreas Bulling
Supervisor:	Florian Strohm, M.Sc.
Commenced:	February 1, 2023
Completed:	August 1, 2023

Abstract

Facial composites have been used by law enforcement agencies to help identify criminals. Initially, these facial composites were sketched by forensic specialists. Later, first software systems emerged to perform this process. But on the one hand, these systems require expert knowledge and, on the other hand, composites are created by focusing on individual parts of the face and then only combining them together in the end. Zaltron et al. presented CG-GAN, a new system that generates composites based on GAN models and in an iterative manner, as recommended by the latest research. However, the system of Zaltron et al. is still difficult to use for non-experts, so we present a new, improved system. We implement a web application for the creation of facial composites. This web application is easy to use and is accessible to all, users do not have to set up the system themselves. Users can also easily edit the features of a face as we have added the ability to adjust the features with range sliders. We also made it possible for users to undo their changes. A user study was conducted to assess whether the usability improvements also contributed to better facial composites.

Contents

List of Figures

List of Tables

List of Listings

List of Algorithms

List of Abbreviations

CelebA CelebFaces Attributes Dataset. 20

GAN Generative Adversarial Network. 17

NASA-TLX NASA Task Load Index. 36

pg-GAN Progressive Growing of GANs. 20

SPA Single-page application. 23

SUS System Usability Scale. 36

1 Introduction

Mental face reconstruction is the process of reconstructing a person's face according to the description of witnesses. These resulting facial composites are used by law enforcement to identify perpetrators. [Man10][WR12][SCGC19]

In the past, these facial composites were drawn by experts according to the description of witnesses. Later, the first software systems for creating software-based facial composites appeared, but this early software still relied on expert knowledge and could only create features of a face if those features were explicitly implemented. So, Zaltron et al. [ZZR20] implement a new way by using machine learning techniques. They used a Generative Adversarial Network (GAN) based model to create high resolution images of person faces. Users are now less depended on expert knowledge when creating facial composites. But there are still some major usability problems which prevent users to use this software trouble-free and create good facial composites. For example, the software is implemented as a jupyter notebook, so users have to set up the software on their own. Moreover, CG-GAN does not provide a good way easily edit features of a face, or with CG-GAN it is not possible to revert changes. With these limitations, it is difficult for users, especially for non-experts, to create good facial composites. In this bachelor thesis, we introduce a new web based system which solves these problems and increase the usability of the system to enable use to create better facial composites. The new system is improved by having a web application, so users do not have to set up the system themselves, and by having a better way to edit features of a face, because with our system we allow users to adjust features with range sliders. We then conduct a user study to compare the usability of the system by Zaltron et al. [ZZR20] with the new implemented web based system. The results show that the created facial composites have the same quality and do not differ significantly. Users enjoyed the new features, such as the sliders, as well as the web application itself. However, the differences in usability between the two systems were not considered significant.

Structure

The thesis has the following structure.

Chapter 2 – Related work: Here we lay the groundwork for this thesis.

Chapter 3 – Web CG-GAN: In this chapter, we introduce related work and the system CG-GAN.

Chapter 4 – New Features: The new features we have implemented to improve usability are presented.

Chapter 5 – Usability improvements: Further usability improvements are shown.

Chapter 6 – Experiments: Here our user study is presented.

Chapter 7 – Discussion: In this chapter, we discuss the results of our user study.

Chapter 8 – Summary and Outlook summarizes our results and outlines future work.

2 Related work

2.1 Facial composite creation

Sketches First facial composites were sketches created by forensic specialists. Forensic specialists drew the facial composites by interviewing eyewitnesses. This procedure focuses on creating individuals parts of a face, e.g. mouth, nose eyes. The created parts will then be combined to form a whole face. Limitations of this method are that the eyewitnesses need to be interviewed and that the forensic specialists need to be trained to create facial composites. Moreover, studies showed that humans perceive faces as a whole, so focusing on individual parts of the face is not ideal. [Man10][WR12][SCGC19]

Feature Software Systems There are also software systems to perform this process. The concept of these feature-based software systems is similar to drawing sketches. Software of this kind also focuses on creating suitable parts of the faces, called features. These features will then get combined to the result face. These tools come with a large dataset that provides different types for each feature, for example, different hairstyles or face shapes. It is then possible to select a suitable variation for each part of the face to create the final composite. The software itself is still operated by experts and depends on interviewing eyewitnesses. Some example software systems are *PRO-fit* and *Faces 4.0*. This method has the same limitations as the sketch method, only that the forensic specialists do not need to be trained to draw facial composites but to use the software. [FBH08] [ZZR20][SCGC19]

Evolving Software Systems As studies have shown, humans perceive faces as a whole, so focusing on individual parts of the face when creating facial composites is not ideal. Therefore, new software systems do not focus on individual face features. Instead, they understand faces holistically and also mutate them in this way. While the other software system or the sketches only select the individual part once, the new systems use an evolutionary and iterative way. The resulting composite is created over multiple iterations. An Example for this kind of software is *EvoFIT*. A drawback of these systems

is that they rely on mathematical functions to evolve and generate features of faces. For this reason, it is difficult to create new features or adjust them according to the requirements of the users. [FHC04] [FBH08][SCGC19]

2.2 CG-GAN

CG-GAN is a new way presented by Zaltron et al. [ZZR20], it creates machine-learning based facial composites in an evolutionary way. CG-GAN is based on Progressive Growing of GANs (pg-GAN), an extended way of a GAN model. The specific pg-GAN used by CG-GAN is trained on the CelebFaces Attributes Dataset (CelebA) dataset. [GPM+14][KALL17][LLWT15][ZZR20]

CG-GAN has two main features that distinguish it from previous systems.

1. Iteration of the generated images. The generated images evolve over several iterations.
2. Feature Locking. CG-GAN has, like the previous systems, features to describe individual parts of the face. But with CG-GAN you can also lock these features so that with new generation these features are not changed. It is also possible to manipulate individual features.

But in this thesis the focus is not on the used models but on improving the usability.

2.2.1 Installation

CG-GAN is implemented as a *jupyter notebook*[1], so every user have to set up his own local environment. As the generative model needs a graphics card to run at an acceptable speed, the setup is not trivial. There are ways that supposedly reduce the effort run CG-GAN like using an online notebook service, as Zaltron et al. did with using Google's *kaggle* to host the notebook[2]. But users must set up this system there anywhere. So, with the use of a web-based system, we easily solve this problem, since every user just needs a browser to access and use the system.

[1] https://jupyter.org/about
[2] https://www.kaggle.com/code/lulliflores/cg-gan/notebook

2.2.2 User Interface

Before we analyze the user interface in more detail, we will give a brief description and explanation of the user interface. The CG-GAN user interface is divided into 4 pages (see Figure 2.1).
The first page is the starting form shown in Figure 2.1a, where the users set their session ID, enter their name and preliminary information about the suspect. There is also a text field to keep notes about the perpetrator. After completing the start form, the session starts.

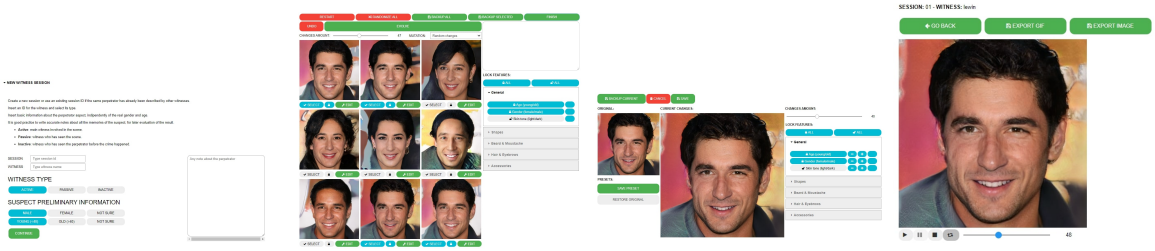
Users are now shown the second page, the main page (see Figure 2.1b). Here the users have nine images which they can select or lock. Selected images are included into generation of the next generation of images. Locked images are kept unchanged for the next generation and are not included in the generation. A combination of both is possible.
The user interface also allows features to be locked. This means that the locked features are not changed for the new generation. With smart lock, not only the current feature is locked, but also all the features that correlate with it.
Users can select the mutation type that determines how features influence the next generations. There are three types of mutations.

1. Random changes. For the next generation, random Gaussian noise is added to the latent vectors.
2. All unlocked features. Randomly increase or decrease a random number of features of a latent vector.
3. One unlocked feature. Randomly increase or decrease a randomly selected single feature of a latent vector.

Moreover, CG-GAN provides a slider to adjust the amount of changes made to the composites as they evolve.

With the third page shown in Figure 2.1c users can edit a single person manually by increasing or decreasing the value of the features, as well as locking the features. There is the possibility to create presets to store intermediate results and to retrieve them later. The edit page provides the same slider as the main page to adjust the amount of changes made to the composites. However, the slider here determines how much a button click changes the feature value. If the slider is set to a higher value, the feature value changes more with each button click than if it is set to a lower value.

When finishing the processing of creating facial composites, users are shown the final page (see Figure 2.1d), where a final animation with some variations of the result image is shown. The final images can also be exported here.



(a) Start Page **(b)** Main Page **(c)** Edit Page **(d)** Final Page

Figure 2.1: User interface of CG-GAN

2.2.3 Problems

We keep the structure of nine images in a 3x3 grid, as it allows the user to keep track of all the images at once. With this structure, the images are stacked next to each other in a compact way, the user can easily compare the images and see the differences between them. We also keep the positioning of the feature lock module.
But we change the placement of all the other buttons. The positions of the buttons cause a bad visual flow, so there is no logical path through the interface. Instead, the user interface has it all mixed up.

Jupyter notebook is mainly an environment for researchers to do scientific computing. *Jupyter notebooks* are not developed to provide a good user interface for users. However, there are libraries such as *IPyWidgets* [3] that provide some interactive widgets. This is also the library used by [ZZR20], but the focus of their project was not on usability, but on developing such a system in itself. A drawback of using this *jupyter notebook* is also that the entire user interface is slow. CG-GAN is slow in terms of displaying the images, as it does not keep the images in a buffer. So if for example the user reverts an action the image needs to be regenerated or in the manual edit view, the presets have no preview. So if you want to use a prior preset, you cannot see a preview of the resulting images with the applied changes.
Moreover, CG-GAN is not responsive, if the user's display is too small they cannot use the system without scrolling horizontally or vertically.

The user interface also lacks physical constraints. Users are often able to perform actions that they should not be able to perform. For example, users can hit the evolve button, although there is no selected mutation type. The user interface then only shows a notice that a mutation type needs to be selected, instead of disabling the button when evolving cannot be performed.

[3] https://ipywidgets.readthedocs.io/en/stable/

3 Web CG-GAN

The idea is to implement a web-based system that uses the CG-GAN model to create facial composites. This web-based system should be easy to use and should not require expert knowledge. A web based system can be used by any user with a web browser and does not require any local installation. This is a big advantage compared to the *jupyter notebook* implementation of CG-GAN and enables a broader user base to use the system. The code of the web based system is available in the respective repository of this thesis[1].

3.1 Architecture

Our system is based on a client-server architecture (see Figure 3.1). The client, our front-end, is used by the users. The server, our back-end, runs the GAN model and generates the respective images. The client communicates with the server with a REST API. Our back-end runs with Python and *Django*[2] as a REST web framework. For the front-end, we use a Single-page application (SPA) with *Vue.js*[3] as the JS framework.

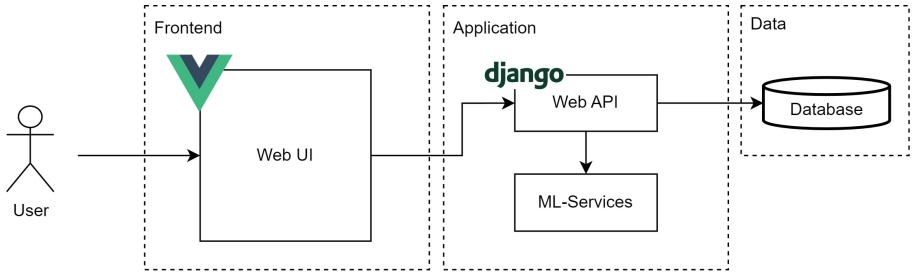


Figure 3.1: Architecture of our web application

[1] https://git.hcics.simtech.uni-stuttgart.de/theses/bsc2023_lewin
[2] https://docs.djangoproject.com/en
[3] https://vuejs.org/

3.2 Back-end

The major purpose of the back-end is to generate images and store vectors of the respective session, so that the front-end can make request to the back-end to manipulate the vectors and get new images.
The back-end runs with python 3.7 due to the compatibility with *TensorFlow* version 1.13[4] which is required by the model used by CG-GAN. As a web framework to build the REST API, we use *Django*.

We use a *SQLite*[5] for providing a simple database. For our prototype and our comparably small user study, we do not need an entire database management system. For a future deployment of the web application, a database management system should be used. The database is needed to store the data required to use the web application and to store the metrics collected for the user study in Chapter 6. Our back-end provides an API endpoint to track the usage of the web app. With each call to the back-end it is possible to track one action, and each action gets stored with a timestamp and its related session. We track all button clicks and all slider drags performed by the users.

3.3 Front-end

The front-end is a web page build as a SPA with *Vue.js* as JS framework. Only the initial page is loaded as an entire new page, later the page gets manipulated by dynamic requests to the back-end. This enables for more fluid usage. [FN98]

Our project mainly focused on the front-end and to make it more usable. The front-end has the same structure as the original CG-GAN implementation. First, the start page, where the user enters his name and basic information about the suspect, our start form is split into multiple steps. Second, the main page, where the user selects the images to be used for the next generation. Third, the edit page, where the user can edit a single person manually. And fourth, the final page, where the user can see the final result and download it.

[4] https://github.com/tensorflow/tensorflow/releases/tag/v1.13.1
[5] https://www.sqlite.org/about.html

3.4 Deployment / Software Engineering

The web application can be easily deployed. It only requires a GPU for the generating model. In addition to the improvements to the user interface, the implementation itself has also been improved. Due to the web application's architecture, the front-end is independent of the back-end and can therefore be extended with additional features without changing the back-end. The same applies for the back-end, because the machine learning model is isolated from the back-end and can be easily replaced with a new model. This enables a more flexible development and deployment of the web application.

4 New Features

In this section, we present the new features we implemented to increase usability. Apart from minor changes, there are four new features for the user interface. The first one is a tag-like system for the features to review and set their lock status (see Section 4.1). The second adds a range slider to each feature to set the value of the feature not only with buttons, but also with sliders (see Section 4.2). As a third change, we added a weighted crossover to enable users to merge faces with a weight (see Section 4.3). Finally, we added a preview to the already existing previews of the presets.

4.1 Tags for feature locking

The user interface of Zaltron et al. [ZZR20] has all the features divided into five groups. As can be seen in Figure 2.1b and Figure 2.1c.

We keep those groups, but instead of using an accordion, we use horizontal tabs (see Figure 4.3). This makes the groups and their items better accessible, as the content of the accordion does not move when opening an accordion tab.

Additionally, we added a tag-like system for the features, as shown in Figure 4.1. This offers a quick overview of the locked tags and allows users to quickly lock or unlock features. We added a search bar with suggestions to make this quick locking possible. Suggested features can be locked by clicking on them or by pressing Enter to lock the first suggested feature. The features can be unlocked by simply clicking on them.

Figure 4.1: Feature tags with suggestions.

4.2 Range slider for feature values

The next new feature, which we implemented, is a new way to change the value of each feature. As the user interface of CG-GAN gives not enough control with its button for the feature values.

The new way of controlling the features is mainly based on a range slider. Each feature gets an additional range slider to the existing increase and decrease buttons. For one reason, we are in line with *Principle of Robustness* and *Principle of Learnability* [Dix03], as we provide a good way to observe the state of the feature and make it more intuitive for first time users.

To improve usability, we avoid making too many calls to change the person. We do this by adding a time-out for each change made by the user. Only if the timeout is reached and no more new changes are incoming, the actual call to the back-end is made. This applies to the increase and decrease button clicks and the slider. With this timeout, we prevent system overload during image generation. So according to the *Principle of Robustness* [Dix03], we maintain a fast response time.

We added a limit for the values of the features to prevent disturbed images. Disturbed images occur when the feature values are set too high and the resulting vector is distorted. See Figure 4.2 for the disturbed image. CG-GAN changes the values of the features by having a vector for each of the features, which it describes. When users now increase or decrease the value of the feature, they are actually changing a factor that multiplies with the vector. The scaled vector is then added to the vector describing the image. The sum of these vectors then describes the new person with changed feature values. But with CG-GAN these factors are not limited, it is possible to click the increase and decrease button an infinite number of times, which then results in a disturbed image.

We keep the procedure by multiplying a factor with the feature vectors, but we limit the size of the factor. Experiments showed that a factor a_f with $-2.5 \leq a_f \leq 2.5$ works best. With this factor, it is possible to force strong changes, but it is so low that it is unlikely to result in distorted images, so that users are no longer able to increase or decrease values limitless.

To highlight the result of a drag operation, we have added a green or red coloring to the slider bar, depending on the direction of the drag. When the value of a feature is increased, then the slider is colored green from its center to the current position of the drag point. Otherwise, it is colored red (see Figure 4.3).

Figure 4.2: Disturbed image. The value of the feature that describes how much the mouth is open is set too high.

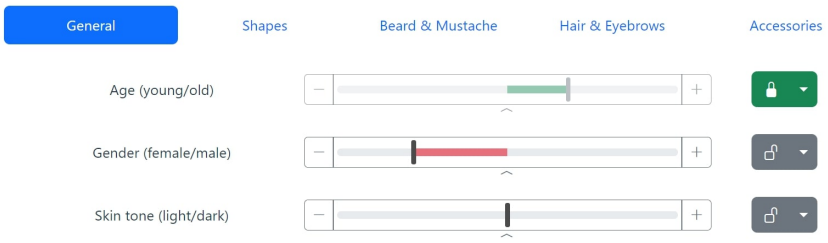


Figure 4.3: Feature range slider. The feature *age* is increased and locked. The value of the feature *gender* is decreased, making the person look more female. The feature *skin tone* is unchanged.

4.3 Weighted Crossover

Figure 4.5 shows the weighted crossover feature. This feature should enable users to merge composites with a weight. For example, if one person looks mostly like the targeted persons but also has similarities with another person. Then, users can try to combine them with a weight.

The first idea was to use a polygon with a person at each corner of it and then create the resulting image based on the position dot and its distance to the corners. However, this feature was only implemented as a prototype, as you can see in Figure 4.4. It is not included in the final version due to its complexity and unclear benefit to users.

The implemented way has only two persons as input and uses a slider between them to set the weight. The closer the slider is moved to a person, the more similar it looks to that person.

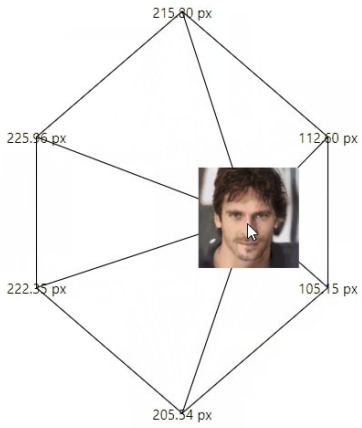


Figure 4.4: Discarded Weighted crossover. Each corner represents a person (they are not displayed for simplicity), but the distance to dragged person is shown. According to the distances from the corners to the dragged persons, the dragged person should be generated. The closer the dragged person is to a corner (another person), the more similar the dragged person is to the person in the corner.

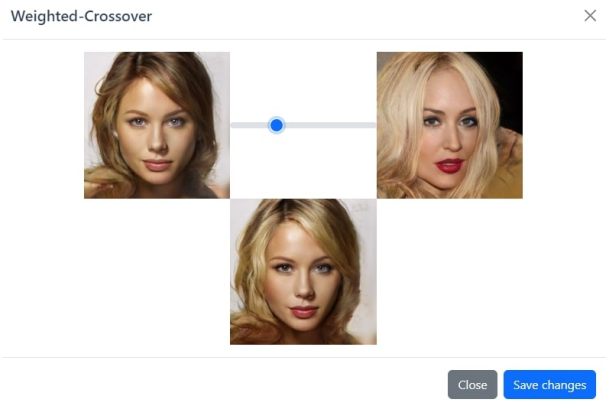
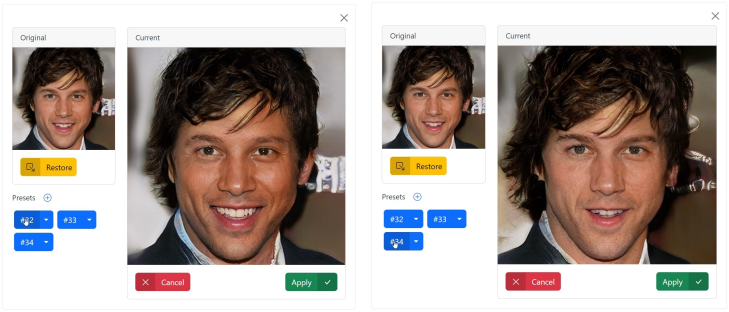


Figure 4.5: Weighted crossover. The slider is moved slightly to the left, so that the combined person looks a little more like the person on the left than the person on the right.



(a) **(b)**

Figure 4.6: Preset hover. The image changes depending on which preset the user hovers over. When the user clicks the button, the preset is selected and the feature settings are loaded.

4.4 Presets Preview

Zaltron et al. [ZZR20] implemented a preset-like system for the manual edit of a person. Where you save your current version of the edited person, to retrieve it later. But with the version of Zaltron et al. every image needs to get regenerated, which takes too much time. We implement a new way to keep the images and the done changes. To enable a preview image when hovering over the respective preset button, so users can quickly check if they want to roll back to a previous edited image. Moreover, we keep the position of the feature sliders, so in contrast to CG-GAN, users can see what changes they have made.

5 Usability improvements

We not only improve usability with new features but also with further improvements. The most obvious improvement is the web application itself because a web application is easily accessible and does not require any technical knowledge to set up.

We improved the form at the beginning according to the rule *design dialogues to yield closure* of the *Eight golden Rules of Interface Design* [SPC+16]. We added a better structure and a progress bar to guide users through the process. With multiple steps, we reduced the complexity of each step and made it easier for future extensions to add new steps.

According to the *Principles of Learnability* [Dix03], we allow users to click on images to select them, not only the small select buttons. Which is more intuitive for first-time users.

Responsive web design Our interface uses Bootstrap, an open-source CSS framework, to provide a responsive web page. CG-GAN, in its *jupyter notebook* is not responsive at all, so we have a big advantage here. Our new interface is usable for all screen sizes. Even tough our web application is not explicitly developed for mobile devices, using it with mobile devices is nevertheless possible.

Reversal of Actions We follow the *Eight golden Rules of Interface Design* [SPC+16] by allowing users to undo their actions at two points. First, we added an undo option to the main page for the evolving process. Users can undo the last evolution step, the previous image is instantly shown because the images of the previous step are saved. Only if users go back more than one step, the images need to be generated again. CG-GAN does not support any kind of undo while evolving.
Second, we improved the possibility to undo changes made to the person while editing them in the manual edit page. As the feature sliders keep the adjusted value, users can see their changes made. They know which features they changed and by what factor, so they can reset the value of a feature to undo the changes. CG-GAN loses the history of changes, users cannot see their changes made, as the buttons does not show how many times they were clicked. So they cannot reset single features, they need to drop all their

changes and restore the original image. The previews mentioned in Section 4.4 also help users to undo changes with our system and CG-GAN. But with our system, they can also undo changes made with the sliders, which is not possible with CG-GAN.

Prevent Errors According to the principles for UI Design by Shneiderman [SPC+16], we prevent errors by limiting the range of the sliders. Sliders have an absolute minimum and maximum position to prevent a disturbed image. Users cannot leave this range with the buttons either. Our system has a preselected mutation type, to prevent errors. Users do not have to select it first or deal with it, which makes it easier for first-time users. Advanced users can then select a specific mutation type for their needs.

In order to simplify the use of our web application, especially for first time users, we hide some non-essential information.

1. Removal of changes amount slider. As we have a slider for each feature, we removed the general *changes amount* slider.
2. Export of latent vectors. CG-GAN allows its users to export the generated latent vectors of persons. We do not implement export functions of these types. As initial users most likely will not export those latent vectors, and it could be unclear for users to distinguish between the normal save and this export function.
3. Hide smart lock of features and weighted crossover. As the correlation of the features is not trivial and not clear for non-expert users, we hid it with a dropdown behind the lock button. The weighted crossover is not a key feature, so we hid it behind the evolve button, not to distract first-time users. Both features can be used by frequent users, to reduce their interactions with the systems to lock multiple features according to the *Eight golden Rules of Interface Design* [SPC+16].
4. Removal of text field for notes. The results of Zaltron et al. [ZZR20] showed that users did not use the text field for notes, so we removed it to reduce the complexity of the user interface.

6 Experiments

6.1 User Study

We conducted a user study to compare our improved user interface with the existing one of Zaltron et al.[ZZR20]. The user study is divided into two parts. First, users create facial composites with both systems. Second, the generated images will be compared to verify if the supposedly improved usability of our system leads to better results. Participants were recruited among family, friends and students. Participants did not receive any compensation for their participation. For the first part of the user study, we had 12 participants. All participants were between 18 and 60 years old (mean$=27.9$, SD$=13.8$). Three of the participants were female and nine were male. For the second part of the user study, we had 22 participants. But we do not have any information about the participants, as this part of the user study was conducted online and no personal information was collected.

6.2 Procedure

Participants should reconstruct two target persons, one with CG-GAN and one using our new UI. The 12 participants used both systems in random order. We have 12 target persons who users need to recreate. Each participant had a different target person for each system. Two participants had the same targets but switched the target for the respective system. As a result, we have 24 composites generated, where two composites generated represent one target, each of the targets where generated with a different system.
As in the user study of Zaltron et al. [ZZR20], the target person is shown to the user for the entire reconstruction time. This helps to avoid the influence of participants not being able to remember the face because we want to focus on evaluating our improved user interface. The target persons were randomly chosen from the CelebA [LLWT15], on which CG-GAN was pre-trained.
Participants got an introduction to the system they were using and got upcoming questions answered. Then they started using the system until they were satisfied with

their result image.
To evaluate the usability of the systems, we used several questionnaires and tracked metrics during their usage.

6.2.1 Questionnaires

The questionnaires were filled in by the users directly after using the system.

1. System Usability Scale (SUS). This questionnaire measures the usability of a system. It outputs a score of 0 to 100. The higher the score, the higher the usability of the system. The participant completed the questionnaire after using each system.
2. Raw NASA Task Load Index (NASA-TLX). A multidimensional questionnaire designed to assess the perceived workload of a user. The final score ranges from 0 to 100. The lower the score, the lower the perceived workload of the task.
3. General questionnaire. We use a non-standardized questionnaire, with our own questions, to evaluate our new features. Some questions are answered with a Likert scale. We use a 5-point Likert scale, where 1 means strongly disagree and 5 means strongly agree. This survey is completed by participants after using both systems.

For the second part of the user study, the evaluation of the generated images, we had 22 participants. All participants who took part in the first part of the user study also participated in the second part. Participants filled out an online questionnaire, where they had to choose the image of the two generated, which they think looks more like the target person. Images were shown in a randomized order.

6.2.2 User Tracking

Additionally, the usage of the applications is tracked. All button clicks made by the user are tracked with a timestamp. For our new system, we implement a simple API endpoint for tracking button click actions. The use of CG-GAN is also tracked; this will provide comparative data. Therefore, we extend the code of CG-GAN with a tracking feature. As in our web version, every button click is logged. But here it is written to a CSV file instead of storing the collected data in a database. All the tracked raw data is available as CSV files in the Git repository of this thesis.

Systems	SUS ↑	Raw NASA-TLX ↓	Time spent ↓ (mins)	Votes for better person ↑ (# comparisons won ↑)
Ours	**94.8**$\pm$ **10.5**	**25.9** $\pm$ **6.6**	12.9 $\pm$ 5.9	9.75 $\pm$ 4.7 **(6 / 12)**
CG-GAN	80.5 $\pm$ 17.4	32.8 $\pm$ 13.0	**11.3** $\pm$ **3.7**	**12.25** $\pm$ **4.7 (6 / 12)**

Table 6.1: Comparison between our System and CG-GAN based on questionnaires and the data tracked during use of the participants. The values shown are the mean and standard deviation of the respective metric. The symbols ↑ and ↓ show if a higher or lower value is better. The better result is written bold. None of the differences are statistically significant, as for all $p > 0.05$.

6.3 Results

For all the collected metrics, we performed a Shapiro-Wilk test to check if the differences between the two systems are normally distributed. If the samples were normally distributed, we performed a paired t-test, otherwise a Wilcoxon signed-rank test was performed. A p-value of $p < 0.05$ was considered significant.

6.3.1 Standardized questionnaire

For the SUS, CG-GAN has an average score of 80.5 and our system has an average score of 94.75. But the difference is not significant, as $t = 2.00, p = 0.071$. With a standard deviation of 17.4 for CG-GAN and 10.5 for our system, the scores for CG-GAN are more spread than the scores for our system. The results of SUS are shown in Table 6.1. For the NASA-TLX, CG-GAN has an average score of 32.8 and our system has an average score of 25.9. But the difference is not significant as well with $t = -1.9, p = 0.084$. As Table 6.1 shows, the standard deviation of CG-GAN is 13.0 and the standard deviation of our system is 6.6. So again, the scores of CG-GAN are more spread than the scores of our system.

6.3.2 General questionnaire

The results of the Likert scales of the general questionnaire are shown in Table 6.2. Most of the participants found our system more pleasant to use, with more than 80% of the participants agreeing or strongly agreeing. For this statement, that is, a mean of 4.1 and a standard deviation of $.086$.

Statements	**Mean ± Std**
The web version is more pleasant to use overall.	4.1 ± 0.86
Rate the efficacy of the feature lock system with the tags.	3.8 ± 0.75
The tags give a helpful overview of the locked features.	3.8 ± 0.69
With the tags feature can be locked faster and easier.	3.7 ± 0.74
Changing the feature values is easier with the sliders than with the +/- buttons.	4.4 ± 1.11
I was able to efficiently achieve the desired result using the sliders.	3.7 ± 0.75

Table 6.2: General questionnaire. The responses are given on a 5-point Likert scale, where 1 means strongly disagree and 5 means strongly agree. The mean and standard deviation of the responses are shown.

More than 60% said they used our tag-based feature lock system. Most of the participants agree that with our tag system, the features can be locked and unlocked more easily. The mean of this statement is 3.8 (SD=0.75). On average, participants agree with a value of 3.8 (SD=0.69) that the tags provide a helpful overview of the currently locked features, so that there is no need to search specific features in the categories.

For the sliders, the users are mostly agreeing or strongly agreeing that changing the feature values is easier with the sliders than with the increase and decrease buttons. Our survey shows that with a mean of 3.7 and a standard deviation of 0.75, the participants are mostly agreeing that with the sliders they could efficiently achieve their desired result.
Moreover, the users stated that with the sliders they could better anticipate the changes of the feature values. And they confirmed that with the sliders they could avoid spamming the buttons to achieve major changes.

The results related to the improved presets are not meaningful as the participants rarely used this feature. But the participants, who used the presets, are mostly agreeing the hover effect simplifies comparing the presets. The situation is similar for the weighted crossover, as this feature was also rarely used by the participants. However, when participants used this feature, they noted that they are not satisfied with this feature because they could not specify which features they want to transfer to other people.

6.3.3 Collected Metrics

Users used our system for an average of 12:56 minutes (SD=5.9) and CG-GAN for an average of 11:17 minutes (SD=3.7). The time difference is not significant, as $t = 1.1, p = 0.29$ and $p > 0.05$.

Users very rarely locked images while evolving. The most of the time, users only selected the person so that the person was locked automatically. The differences between the two systems are not significant, for both locking and selecting. For CG-GAN, users locked on average 4.4 times (SD=14.1) and selected on average 22.3 times (SD=13.6). For our system, users locked on average 0.8 times (SD=1.1) and selected on average 15.0 times (SD=8.1). But for both systems the difference between locking and selecting is significant, as $t = -5.0, p = 0.0004$ for CG-GAN and $t = -5.6, p = 0.0002$ for our system.

As stated in the results of the general questionnaire, users changed the features values with our system less often than with CG-GAN. The total number of calls made to the pg-GAN model to change feature values with CG-GAN was $1,122$ and with our system there were 878 calls. But as our system has a timeout feature to prevent too many calls in a short time, we only generated after a series of calls. Therefore, the images were only generated 506 times, that is, a decrease of 54.9% compared to CG-GAN. This leads to far less delay when generating images, which then results in increased usability.
However, the difference between the two systems and number of changes made was not significant, as $t = 13.0, p = 0.075$.

The number of times that users used the buttons to change the feature values is for CG-GAN on average 93.5 times (SD=67.9) and for our system on average 73.2 times (SD=177.2). The difference is not significant, as $t = 13.0, p = 0.075$.
The reset button, which resets all feature values to zero, was only used twice by one user.

Another interesting metric is the number of times users left the edit page without saving and canceled their editing. First of all, for CG-GAN on average 1.8 times (SD=0.90) and for our system on average 4.3 times (SD=3.33) entered the edit page. The difference between the two systems is significant, as $t = 0.0, p = 0.004$. But our system users left the edit page without saving on average 2.1 times (SD=3.4), while the users of CG-GAN left the edit page without saving on average 0.5 times (SD=0.87). This difference is not significant, as $t = 4.0, p = 0.17$. On average, 0.28 (SD=0.31) of the edits were canceled by users of our system, and 0.21 (SD =0.32) of the edits were canceled by users of CG-GAN. This difference is again not significant, as $t = 4.0, p = 0.17$.

6.3.4 Observations

We did not track all actions such as mouse movements or clicks on disabled buttons and sliders. However, in the following section, we describe further observations we made during the user study.

We observed users trying to click or drag the slider, even though the respective feature was locked and therefore could not be used. With CG-GAN, buttons were still clickable, but then showed a warning message that the feature was locked. With our system, the buttons and sliders were disabled and grayed out, but few users still tried to click or drag them.

When users used the web version first and then the notebook version, they often clicked on the image instead of the select button, but CG-GAN only supports the button clicks. This shows that users adapted the way of clicking on the image instead of the button.

Participants rarely used the preset feature on either system, despite the fact that our system has improved the use of presets. But, with the slider changes, our system offers a better way to keep track of the feature changes, so that presets may become obsolete.

With CG-GAN, users spam the plus button to significantly increase a feature value, the generation of all the single images then takes some time. But users did not wait for the final change and kept clicking the button. The resulting image has then changed too much, so users then spammed the decrease button to undo the changes. Users could have increased the amount of changes, but almost none of them used this feature. This did not happen with our UI, users simply dragged the slider to the desired position. Due to our implementation, images were generated only once, even when buttons were quickly clicked multiple times. Limiting the value of the feature also helped prevent this effect of users forcing too much change by spamming the increase or decrease buttons.

Users often did not select a mutation type before clicking the evolve button, resulting in a warning message saying that no mutation type was selected. Although the message was clearly about the problem and its solution, users were confused and asked what they had done wrong. Our UI made it easier to evolve by having pre-selected elements for the mutation type selection, which resulted in users rarely changing the mutation type.

There were also some problems with the categories chosen for the features. Users often searched for features in the wrong category; although we added our tags with the search bar, users have not adopted the feature enough. This could also be due to the fact that with the feature tags, features could only be locked and unlocked. But when using the edit page, users could not search for features when they wanted to change the value of a feature.
A further problem was that the categories of the features are not equally distributed, so some categories have more features than others. With too many features in a category, users had to scroll to find the feature they wanted. If they then changed the feature value, the resulting images were out of the view and users had to scroll back to see the result.

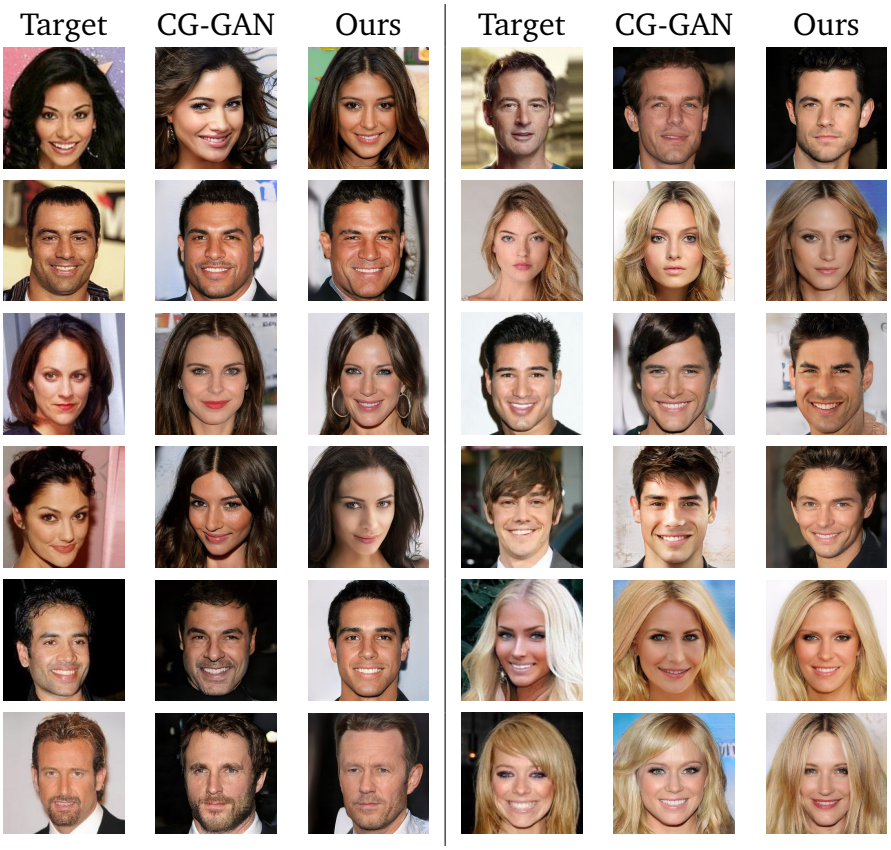


Figure 6.1: Generated composites. The *target* columns show the target person, which the participants had to recreate. The *CG-GAN* columns show the generated composites with CG-GAN and the *Ours* columns show the generated composites with our system.

As the results showed in the user study by Zaltron et al. [ZZR20] users missed the opportunity to change the color of the eyes or transfer a single feature to another person. Participants did not mention this in the surveys, but some asked for this feature while creating the composites.

6.3.5 Evaluation of generated composites

Composites generated by our system had an average of 9.8 (SD=4.7) votes that they looked more like the target person than composites generated by CG-GAN. In contrast, the composites generated by CG-GAN had an average of 12.3 (SD=4.7) votes. The difference between the evaluated images is not significant, as $t = -0.9, p = 0.39$. However, in the end, half of the images that were selected as being better were generated by our system, and the other half were generated by CG-GAN.

7 Discussion

In this chapter, we discuss the results of our user study. None of our results are statistically significant, so we cannot prove our hypothesis that we have improved the usability of the system and that we are creating better facial composites as a result.

One reason for this could be our small sample size for the user study, especially for the first part of the user study when using both systems. For the second part of the user study, the evaluation of the generated facial composites, we had a larger sample size.

An important factor of CG-GAN was missing in our user study. The participants did not need to set up the notebook system for themselves; they got it set up and just needed to use it. This could be one reason why users did not have so much trouble with the system and why the results of NASA-TLX are not significantly different from our system. Therefore, this advantage of our system was not included in the user study, but this was necessary because we wanted to compare the usability of the two systems in the first place. Another reason could be that most of the participants were computer science students and, therefore, are more experienced with using new and unknown software.

As mentioned in Section 6.3.3 the reset button for the feature value slider was almost never used. For one reason, it could be that users did not notice the reset button. Or for another reason, it could be that users did not need to reset the feature value slider and simply dragged the slider back to the original value. In any case, the reset button should be kept, as it is a fundamental feature, but perhaps it needs to be made more visible. The tracking data also shows that users may be more efficient at adjusting feature values with our system, so they do not need to make as many changes.

Our attempt to visualize disabled buttons or sliders with grayed out colors was not effective enough, as few users still tried to use them. Since this was only our observation, and we did not track this, it is not proven that this is a significant problem. However, it may be necessary to make it more obvious to the user that certain buttons are disabled.

The fact that users more often left the edit page without saving their changes with our system is interesting because it contradicts with the other results of the user study, where users enjoyed using the slider to manually edit the features. So, it could be that either users could not produce their desired result or that they just wanted to experiment more, as it is easier with the sliders.

Features such as the tag-based feature overview, the smart lock, and the weighted crossover were less popular with users. This could be a sign that the inexperienced users only focused on the most important features, as our system allowed them to do. However, the weighted crossover is not very useful in its current state and has not been adapted by users. Users usually do not want to combine two composites with weight, but they want to transfer specific features from one person to another.

The focus of the software was on the usability of the system, not on software performance. The web application has some more latency due to the communication between the client and the server. The web application would need to be optimized to reduce the latency, to further improve the response time of the system.

8 Summary and Outlook

In this chapter, we summarize the results of our work and give an outlook on future work.

8.1 Summary

We were not able to show that users create better face composites with our system, but the composites created are not worse either. The improvement in usability was also not shown to be significant. However, the results are still promising and show that users are enjoying the new features, such as the range slider for changing the feature values, and the web application itself. Other improvements, such as the tag-based feature overview or the weighted crossover, were less popular with users.

8.2 Outlook

Future work is needed to investigate whether experienced users use this system differently, if they use advanced features, such as the weighted crossover or the smart lock, more often. It could also be investigated whether adding an expert mode would provide further usability improvements, where experienced users could optionally disable the feature value limits so that they are not constrained by the system to set high values for features. It is then the user's responsibility not to generate distorted images, as the system no longer prevents user error.

For problems with the categorization of the features, different solutions could be implemented. One solution could be just to use different categories, to distribute the features more evenly and to make it easier for users to find the right category. Another solution could be to add a search or filter function, so that users can search and filter for the feature they want to change, so that the search function is not exclusively for the tags and locking and unlocking of features, but also for the features themselves. A third solution could be to add favorites, so that users can mark features as favorites and then only see their favorites. Another solution is needed when users scroll down to the

bottom of the page looking for certain features and no longer see the resulting image. An easy fix could be to make the resulting image stay at the top of the screen or simply make the features scrollable so that users can scroll through the features and always see the resulting image.

Since most users did not use the lock and select feature while evolving the image, it might be useful to show only the select button to first-time users.

With more efficient models or more GPU power, it would be possible to implement a live preview for certain actions like dragging the sliders. Where the image is updated in real time and the change of the feature is shown, not only after releasing the slider. This would improve the interactions with the users.

In the future, the deficits regarding image generation could be overcome by using a new and better model. For example, *StyleGAN* could be used. With *StyleGAN*, for example, it is possible to change someone's eye color. With future work, it is possible to implement the ability to transfer a feature, a specific part of the face, from one person to another. *StyleCLIP*, based on *StyleGAN*, supports a text-driven manipulation of images by text input. In a future version, this could be added as a feature to not only manipulate the image by increasing or decreasing the feature value with buttons or sliders, but also by text descriptions. [KLA19] [PWS+21]

Our web application is only a prototype and implemented for a small user study and needs further development before it can be used in production. But in future this web application could be deployed. When deploying this system publicly and with a larger user base, there could be more metrics collected, to further improve the UI.

Bibliography

[Dix03] A. Dix. *Human-computer interaction*. Pearson Education, 2003 (cit. on pp. 28, 33).

[FBH08] C. D. Frowd, V. Bruce, P. J. Hancock. "Changing the face of criminal identification." In: *The Psychologist* 21.8 (2008), pp. 668–672 (cit. on pp. 19, 20).

[FHC04] C. D. Frowd, P. J. Hancock, D. Carson. "EvoFIT: A holistic, evolutionary facial imaging technique for creating composites." In: *ACM Transactions on applied perception (TAP)* 1.1 (2004), pp. 19–39 (cit. on p. 20).

[FN98] D. Flanagan, G. M. Novak. *Java-Script: The Definitive Guide*. 1998 (cit. on p. 24).

[GPM+14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio. "Generative adversarial nets." In: *Advances in neural information processing systems* 27 (2014) (cit. on p. 20).

[KALL17] T. Karras, T. Aila, S. Laine, J. Lehtinen. "Progressive growing of gans for improved quality, stability, and variation." In: *arXiv preprint arXiv:1710.10196* (2017) (cit. on p. 20).

[KLA19] T. Karras, S. Laine, T. Aila. "A style-based generator architecture for generative adversarial networks." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 4401–4410 (cit. on p. 46).

[LLWT15] Z. Liu, P. Luo, X. Wang, X. Tang. "Deep Learning Face Attributes in the Wild." In: *Proceedings of International Conference on Computer Vision (ICCV)*. Dec. 2015 (cit. on pp. 20, 35).

[Man10] S. Mancusi. *The police composite sketch*. Springer Science & Business Media, 2010 (cit. on pp. 17, 19).

[PWS+21] O. Patashnik, Z. Wu, E. Shechtman, D. Cohen-Or, D. Lischinski. "Styleclip: Text-driven manipulation of stylegan imagery." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 2085–2094 (cit. on p. 46).

[SCGC19] C. N. Stephan, J. M. Caple, P. Guyomarc'h, P. Claes. "An overview of the latest developments in facial imaging." In: *Forensic sciences research* 4.1 (2019), pp. 10–28 (cit. on pp. 17, 19, 20).

[SPC+16] B. Shneiderman, C. Plaisant, M. S. Cohen, S. Jacobs, N. Elmqvist, N. Diakopoulos. *Designing the user interface: strategies for effective human-computer interaction*. Pearson, 2016 (cit. on pp. 33, 34).

[WR12] C. Wilkinson, C. Rynn. *Craniofacial identification*. Cambridge University Press, 2012 (cit. on pp. 17, 19).

[ZZR20] N. Zaltron, L. Zurlo, S. Risi. "CG-GAN: An Interactive Evolutionary GAN-Based Approach for Facial Composite Generation." In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.03 (Apr. 2020), pp. 2544–2551. DOI: 10.1609/aaai.v34i03.5637. URL: https://ojs.aaai.org/index.php/AAAI/article/view/5637 (cit. on pp. 17, 19, 20, 22, 27, 31, 34, 35, 41).

All links were last followed on July 17, 2023.