Matching human faces to opposite gendered anime faces

Nick van der Merwe
nick.vandermerwe@griffithuni.edu.au - s5151332 - 2812ICT

Abstract
In this piece a system that ranks the similarity between human faces and anime faces was designed. It was built on comparing the eye, hair and edges of the provided image to the other images. It was specifically done avoiding data heavy approaches outside of pretrained cascade recognition systems. It resulted in anime characters hair being identified correctly 36% of the time, and their eyes 51% of the time. Human faces had the correct hair colour 59% of the time and 43% of the time had the correct eye colour. Due to the inherent nature of comparing opposite gendered anime faces to people, how accurate the final result is subjective.
Keywords: facial recognition, cartoon recognition

1. Introduction

In modern society it has been proven that marriage rates are lower than ever(Curtin SC, 2020), and its also proven that there exists a correlation between a humans appearance and their partners(Ducharm, 2019). Recently, due to zoomer culture the world of waifus has been rapidly expanding - this is where people consider a fictional person as their partner. These facts open the demand for a system that finds a waifu for a person given their physical appearance.

In terms of how this was approached, the main aspects of what makes an anime character similar to a person were considered. This is primarily eye colour, hair colour, and to a degree the edges of their facial features. Due to the simplified art style, if those are reasonably matched then majority of the time a person would consider it a match. This will be approached without any neural networks or deep learning systems, and purely an approach focusing on the perceptual computing side of system. The overarching goal is to explore methods to analyse both human and cartoon faces in a broad take.

The rest of the report is organised as 2. is the previous works, 3. is the technical approach, 4. contains the results, 5. contains the conclusion and 6. includes the references.

2. Previous works

The major categories that are related to this are cartoon face recognition, human facial recognition and image similarity ranking systems.

The most related paper to these methods is *"Face Detection and Face Recognition of Cartoon Characters Using Feature Extraction"* (Takayama et al., 2012). In this paper they primarily use jaw lines as the defining aspect of animated faces, then use the k-means method to extract colours and segment features of the face out. In the end they achieve an F-measure of 0.516, which means its catering on the edge of being considered a poor rating. It is to be expected that cartoon faces are especially difficult to categorise due to how much variance there is in comparison to human faces. Its also a part of recognition systems that is not often considered.

Another flagship paper, in cartoon face recognition is *Bringing Cartoons to Life: Towards Improved Cartoon Face Detection and Recognition Systems*, however, it has a focus on training a neural network which is where the approach differs. Through this they achieve 0.649 F-Measure (Jha et al., 2018)

Currently, majority of cutting edge facial recognition systems are behind companies that are privatised. However, even if we look at older research from Google, their paper from 2015, *FaceNet: A unified Embedding for Face Recognition and Clustering*, achieved 99.63% accuracy. In comparison to the current f-score of cartoon faces, that difference is visible. Humans, in comparison are at 98.52% accuracy - which means the problem was effectively solved five years ago as of this paper. This system Google primarily revolves around deep convolutional networks training with triplet loss and triplet selection. (Schroff et al., 2015)

In terms of ranking images by their similarity, the main approach is deep learning. Across contexts and if the system is searching through millions of images, basic approaches such as SIFT simply do not function well enough. In 2014 a flagship paper was released, *"Learning Fine-grained Image Similarity with Deep Ranking"*. In it, they explore training a major deep neural network and focus on triplet sampling and layer sampling. This approach results in a 85.7% precision. Considering this system is not trained towards a specific style of image and that the first result is related to the image 85.7% shows how accurate deep neural network models can become (Wang et al., 2014).

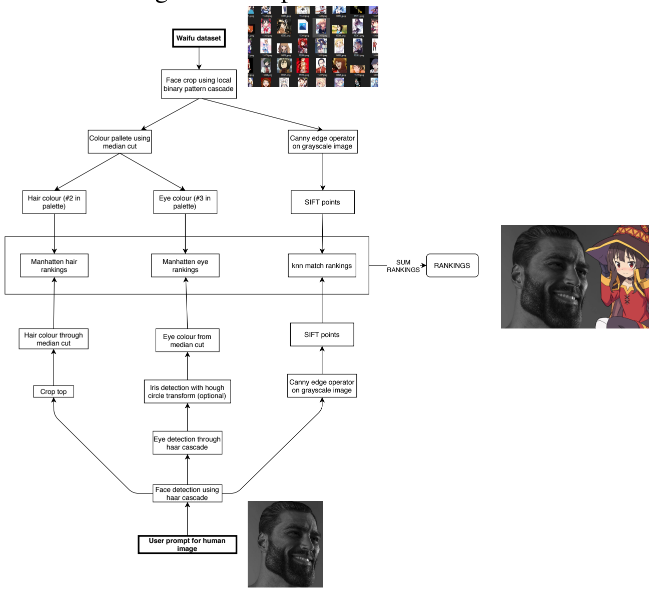
3. Technical Approach

The overall model is to have a database of animated faces and preprocess these into hair colour, eye colour and SIFT points. Once these are preprocessed into a compact file, the user opens it with their image of a human. A visual model can be seen in figure 1, and next we will step through how everything works.
Since the model is not linear, exploring each section in order is not an option. Here each part will be explained and how they relate to each other, then each major function will have a subsection dedicated to it. The algorithms are cascades in section 3.1., canny operator in section 3.2., SIFT in section 3.3., hough circles in 3.4. and finally median cut colour quantization in 3.5..

Our images are fundamentally split into two groups - animated and real life. These each have separate properties, and a human face needs more effort to extract these than an animated face.

Our goal with anime faces is to process them into the

Figure 1: Simplified overall model


Waifu dataset
Face crop using local binary pattern cascade
Colour pallete using median cut
Canny edge operator on grayscale image
Hair colour (#2 in palette)
Eye colour (#3 in palette)
SIFT points
Manhatten hair rankings
Manhatten eye rankings
knn match rankings
SUM RANKINGS
RANKINGS
Hair colour through median cut
Eye colour from median cut
SIFT points
Crop top
Iris detection with hough circle transform (optional)
Canny edge operator on grayscale image
Eye detection through haar cascade
Face detection using haar cascade
User prompt for human image

three key pieces of information we need, so this side is given a file of faces to process in the first stage. For each anime face we start with cropping it with a pretrained local binary pattern cascade from `https://github.com/nagadomi/lbpcascade_animeface`. Once its cropped, we run the median cut algorithm, which removes the largest groups of colours from the image and we assume the first one is skin (so its ignored), the second is hair and the third is eye colour - these are saved as RGB values. Finally, we filter the gray scale image with a canny operator and save the SIFT points. This is because the main focus that defines the animated face here are the edges. These classes are saved to a large pickle file that contains the file name and each of these variables to search through in the next step.

Our next major step is to grab the user input of what face they want to search for waifus with. On this face, we need to crop out the actual face and inside of that crop the top of face and the eyes. The eyes we can use another opencv pretrained cascade, $haarcascade_eye.xm$, but the top of the head we simply assume that the top 10% of the face contains hair. On the eye crop we can try to filter it further by using hough circle transformations, which labels circles. This tends to fail if the image has their eyes half open, so it skips if a circle is not found. Then on our hair and eye crops, we run the median cut function and return the most dominant colour bucket in the crops as their respective colour.

To grab the person's SIFT key points and descriptors its the same algorithm as the animated face - we run a canny operation on the face then sift on it.

Now that we have all of the data of each face, the remaining problem is to rank each one. At this point the weight of each variable was not adjusted, and the colour, hair and SIFT rankings were each worth the same amount. Essentially the hair and eye colours were sorted by their euclidean distance from the human face's eyes and hair, using euclidean as a measure because it gives a reasonable human interpretation of how far the colour is (Riemersma,). Once that's complete, SIFT points were compared using k-nearest neighbours match (the closest neighbour in the corresponding image is compared to it), and if the distance between two points was within a 25% margin, it was marked as a good point. Our rating of the quality of the SIFT the fraction of good matches out of the total key points.

Now that we have all three of the rankings, we can combine these. The final ranking takes the position of each image in their trees and sums the position to the final tree. For instance, if an image came 27th in hair, 54th in eyes and 112th in SIFT its score would be $27 + 54 + 112 = 193$. Using this score for each image, the minimal ones are the highest related to the original. This is then returned to the user that requested it.

3.1. Crops with cascades

Cascades are a system that are trained through a dataset that contains labelled positive and negative cases to recognise patterns (Gama and Brazdil, 2000). Depending on the type of cascade, this part differs - here we use haar and local binary pattern for different parts of recognition.

Essentially haar functions by extracting edge features, line features and four-rectangle features from images using kernels. However, if a cascade system was trained purely through kernel filters it would end up costing a lot of computational power so it is assisted with adaboost. This is a style of boosting; each feature is ran on a dataset and find the best thresholds for parameters(Viola and Jones,).

The functional difference between haar and local binary patterns (LBP), is that LBP functions through calculating the differences in surrounding images and its difference is compared to a threshold(Ojala et al., 1996). The overall difference between these ends up being that LBP tends to be significantly faster (2.5 times faster), but haar is more accurate (89% vs 84.7%)(Kadir et al., 2014).

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c) \times 2^p$$

$$s((x)) = \left\{ \begin{array}{l} 1, if x \geq 0 \\ 0, otherwise \end{array} \right\}$$

Figure 2: Local Binary Pattern equation

3.2. Canny operator

The canny operator handles extracting edges from images. Considering that anime images are largely dependant on their edges, this is mostly useful to simplify the human face for the SIFT operation. In terms of how it works, it has four major steps: filter with horizontal and vertical Gaussian filters, calculate the orientations and magnitudes of the gradients and clean it with non-maximum suppression and lastly hysteresis (Canny, 1986).

Horizontal and vertical Gaussian gradients are reasonably simple - from a given pixel, calculate the difference in

$$\theta = tan^{-}1(\frac{\partial f}{\partial y}/\frac{\partial f}{\partial x})$$

Figure 3: Orientation of points formula

colour to the given direction and multiply along it with a Gaussian kernel. To generate the gradient magnitude, we simply have to put these two gradient images into a euclidean distance formula, and the orientations comes from an inverse tan equation

Once we have those, it is processed through non-maxima suppression which thins down the thicker edges by interpolating the line along the gradient direction. Then we can filter through hysteresis, which removes weak lines and connects drop out lines.

3.3. SIFT points

Once we have our canny filtered image from either our human face or anime face, we need to process it into SIFT - Scale-Invariant Feature Transform. This consists of four major steps: Scale-space feature detection, keypoint localisation, orientation assignment and keypoint descriptors. Essentially, it takes an image in and returns the key points and their descriptors for k-nearest neighbours matching (the closest points in the corresponding image are compared to it) (Lowe, 1999).

The idea of the scale-space feature detection is that the convolution operator is ran to generate multiple blurred images. Through this a scale invariance is created, and the maximas and minimas of the difference of gradients formula is picked in each scale using the equation in figure 4. Once they have been found, they are compared between each scale, and if they are neighbours between scales, they are added as options.

$$D(x, y, \sigma) = L(x, y, k_i\sigma) - L(x, y, k_j\sigma)$$

$$L(x, y, k\sigma) = G(x, y, k\sigma) \times I(x, y)$$

Figure 4: Difference of Gaussians used for SIFT. L = convolution of image,

The rest of the steps that follow flesh out those key points. The localisation step reduces the number of key points in the overall image by considering the nearby pixels of each point and deletes unstable ones. Orientations label each with a gradeint direction for more detail to add invariance from lighting changings, camera angles and related factors. This is achieved by factoring in the local region's gradient histogram.

To implement SIFT in the code, opencv's $xfeatures2d.SIFT_create()$ function was used and $detectAndCompute()$ to generate the key points and descriptors.

3.4. Hough Circles

The hough circle transform is an algorithm to find and label circles in an image. This is used after the eye is cropped to label the iris. The idea here is that the iris of the eye is a distinct circle, so if the image is ran through a median blur and then processed with a hough circle transform it can be labelled more distinctly. However, in practice it tended to fail reasonably often. As a result, if it fails to find any circles the program simply uses the original eye crop in the colour palette stage.

The way that hough circle transform works is that divides the image into buckets and makes an accumulator matrix for each portion, which counts how many potential circles are in that portion. It then votes, in the canny image, how many edges agree with the circle in the accumulator matrix and returns the maximum voted circle (Illingworth and Kittler, 1987).

In practice, this was done utilising opencv's $cv2.HoughCircles()$ function.

3.5. Median Cut algorithm

Median cut is a colour quantization approach which reduces the number of colours in an image and returns how common each is. This is used at different stages in the human face and anime faces due to the inherent difference between cartoon characters and humans - humans have more texture. In an anime face, its accurate to grab the entire face and rank the first colour this outputs as the skin, the second the hair and the third the eyes.

However, in a human face its suspect to lighting environments, skin tone uneveness and multiple other factors which can introduce complications. As such, the other crops were done to make this more accurate.

In terms of how it functions, its reasonabley simple. It searches for the smallest box in the image which contains every possible colour, and then sorts them. It splits the box at the median colour, then does the same recursively between the groups until there are an equal number or more colours than requested. As a result, the first rank has the biggest group of related pixels together, the second the second and so on (Segenchuk, 1997).

4. Experiments

Since we are comparing a human face to a animated face, its subjective whether they are actually the correct match. On top of this, one of the primary algorithms utilised, SIFT, already has a lot of data on how well it works. As a result, the main variables that we can check whether they are correct is just the hair and eye colour detection in characters and people. We can also display a couple of human faces with the related anime faces as a review on the system.

In terms of the datasets used, the human faces are from `https://www.kaggle.com/ashwingupta3012/human-faces` and the anime faces are from `https://www.kaggle.com/splcher/animefacedataset`

4.1. Colour selection

To rate the system's colour detection, a function was made that takes in a folder of images, picks 100 out of them and prints a rectangle of the variables alongside the picture of the person/anime character. Then it was judged by hand whether the correct colours were picked.

Faces	Correct hair colour	Correct eye colour
Anime faces	36%	51%
Human faces	59%	43%

While complimenting this with the precision, recall and f-measure would be preferable, due to there being multiple colours of hair sections such as false positive do not exist - i.e. we are not measuring if a person has blue hair.

4.2. Samples

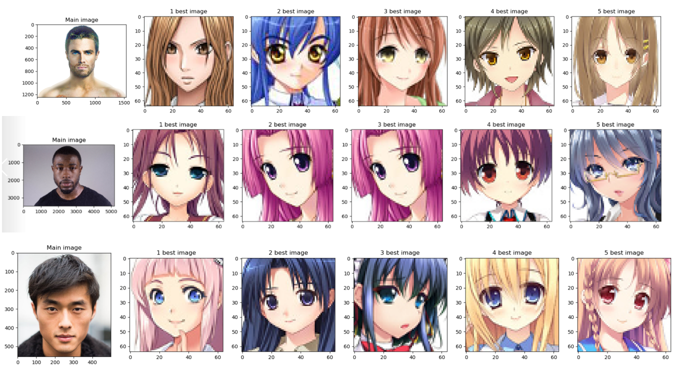


Figure 5: Example outputs from the system

As visible in figure 5, there are duplicates in the dataset, and this managed to find them. However, overall it is not a bad output. The major face here that went wrong is the second person outputted pink haired anime faces when his hair and eyes are black for all five. Presumably the system picked up a slight pink tone from the darkness of his hair, because it also occurs in the third image.

The first image is interesting because the top character matches his hair colour, then the second one is the exact inverse of his hair colour - blue hair and hazel eyes.

5. Conclusion

Overall, its visible that this is not an exact algorithm. However, this approach (aside from face and eye detection) used no data-dependant models such as neural networks, which require a large chunk of data. A large chunk of the functions are dependant on variables such as hough circles and can be optimised.

6. References

Canny, J. (1986). A computational approach to edge detection. https://doi.org/10.1109/tpami.1986.4767851.

Curtin SC, S. P. (2020). Marriage rates in the united states. https://www.cdc.gov/nchs/data/hestat/marriage_rate_2018/marriage_rate_2018.pdf.

Ducharm, J. (2019). Why do so many couples look alike? here's the psychology behind the weird phenomenon. https://time.com/5553817/couples-who-look-alike/.

Gama, J. and Brazdil, P. (2000). https://doi.org/10.1023/a:1007652114878.

Illingworth, J. and Kittler, J. (1987). The adaptive hough transform.

Jha, S., Agarwal, N., and Agarwal, S. (2018). Towards improved cartoon face detection and recognition systems. http://arxiv.org/abs/1804.01753.

Kadir, K., Kamaruddin, M. K., Nasir, H., Safie, S. I., and Bakti, Z. A. K. (2014). A comparative study between LBP and haar-like features for face detection using OpenCV. https://doi.org/10.1109/ice2t.2014.7006273.

Lowe, D. (1999). Object recognition from local scale-invariant features. https://doi.org/10.1109/iccv.1999.790410.

Ojala, T., Pietikäinen, M., and Harwood, D. (1996). A comparative study of texture measures with classification based on featured distributions. https://doi.org/10.1016/0031-3203(95)00067-4.

Riemersma, T. Colour metric. https://www.compuphase.com/cmetric.htm.

Schroff, F., Kalenichenko, D., and Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. https://doi.org/10.1109/cvpr.2015.7298682.

Segenchuk, S. (1997). An overview of color quantization techniques. https://web.cs.wpi.edu/~matt/courses/cs563/talks/color_quant/CQindex.html.

Takayama, K., Johan, H., and Nishita, T. (2012). Face detection and face recognition of cartoon characters using feature extraction. http://iieej.org/trans/IEVC/IEVC2012/PDF/4B-1.pdf.

Viola, P. and Jones, M. Rapid object detection using a boosted cascade of simple features. https://doi.org/10.1109/cvpr.2001.990517.

Wang, J., Song, Y., Leung, T., Rosenberg, C., Wang, J., Philbin, J., Chen, B., and Wu, Y. (2014). Learning fine-grained image similarity with deep ranking. https://doi.org/10.1109/cvpr.2014.180.