Ashutosh Kumar

2022AIM1001

Report PA2

Image Morphing with Python

Introduction

Image morphing is a technique in computer graphics used to create a smooth transition between two images. One of the critical steps in creating a morphed image is to find corresponding points between two images. Delaunay triangulation is a standard method for finding these corresponding points. In this report, we will explain the steps involved in face morphing using Delaunay triangulation.

This code morphs two images of faces together and creates a GIF of the morphing process. Here are the steps involved in the code:

Steps in Face Morphing using Delaunay Triangulation

1. Load the images "morgan.jpg" and "ben.jpg" using the OpenCV library's imread() function and store them in the variables img2 and img1, respectively.
2. Use Dlib's facial detection library to detect the faces in each image.
3. Find the facial tile points for each image using a function called find_facial_tilepts(). These points correspond to critical facial features, such as the corners of the eyes, nose, and mouth.
4. Mark the coordinates of the facial tile points and the four corner points on each image using the mark_coord() function.
5. Create an empty list called ind_pnt_tri to store the triangle indices for each intermediate frame.
6. Use the Delaunay triangulation algorithm to triangulate the facial tile points in one of the images, img1.

7. Loop through each triangle in the triangulation and add a tuple of the indices of its points to ind_pnt_tri.
8. Initialize the number of frames to 30 and create an empty list called morph_list_img to store the morphed images.
9. Loop through each frame and create an intermediate frame by interpolating between corresponding points in both images.
10. For each triangle in the triangulation, extract the indices of the points that form the triangle, as well as their coordinates in the original images and in the intermediate frame.
11. Apply an affine transformation to the triangle using the Affine_transform() function and add it to the morphed image for the current frame.
12. Append the morphed image to morph_list_img for the current frame.
13. Convert the original images and the morphed images to RGB format and create a GIF animation using the Image and Imageio libraries.
14. Save the GIF animation as "PartA/B.gif".

    Overall, the code uses a combination of facial detection, triangulation, and affine transformations to morph two images of faces together and create a visual representation of the transition.

In conclusion, face morphing using Delaunay triangulation is a powerful technique that can create a smooth transition between two images. The process involves detecting faces, obtaining landmark points, applying Delaunay triangulation, obtaining a one-to-one mapping, and applying affine transformations to create the morphed image. With advances in machine learning and computer vision, this technique has become more accessible and is now used in various applications, including digital entertainment and medical imaging.

How to run the code(Readme):

This is a Python program that allows you to morph two facial images together. The program uses OpenCV and Dlib libraries to detect facial landmarks and create a morphing effect between two input images.

Running the Program on Google Colab

Prerequisites

- A Google collab account
- Uploading the required images and data files in the current directory, i.e., content/

Instructions

1. Upload two facial images of the same size in the content/ folder of the current directory. Example images are given in the zip file: morgan.jpg, ben.jpg.
2. For Part A (user input from file), upload a text file named tile_points_72.txt or your own input file with the same name and run all the cells one by one. The output is PartA.gif.
3. For Part B (automatic tile point detection), download the shape_predictor_68_face_landmarks.dat file by running the second cell in the notebook, then comment it out, and run all the cells. Alternatively, if you have already downloaded the file, you can skip the second cell and run all the cells directly. The output is PartB.gif.

Running the Program as a Python File

Prerequisites

- Python 3.x
- OpenCV, Dlib and other libraries installed

- shape_predictor_68_face_landmarks.dat file in your working directory

Instructions

1 Upload two facial images of the same size in the current directory. Example images are given in the zip file: morgan.jpg, ben.jpg.
2 For Part A (user input from file), upload a text file named tile_points_72.txt or your input file with the same name and run the Python code. When prompted for input, press A or a for Part A. The output is PartA.gif.
For Part B (automatic tile point detection), simply run the Python code. When prompted for input, press B or b. The output is PartB.gif.