VIETNAM NATIONAL UNIVERSITY OF HO CHI MINH CITY

HO CHI MINH UNIVERSITY OF TECHNOLOGY

FACULTY OF COMPUTER SCIENCE AND ENGINEERING

FINAL PROJECT REPORT

MACHINE LEARNING ASSIGNMENT

TOPIC: CONTROL THE MOUSE CURSOR WITH FACIAL MOVEMENT

Advisor: Nguyen Duc Dung

Student: Nguyen Ton Minh - 2052600

Dinh Truc Tam - 2053415

Ho Chi Minh City – 2023

AIM OF THE PROJECT

The primary problem the project addresses is the need for alternative methods of human-computer interaction, specifically for individuals who may not be able to use traditional input devices such as a mouse or keyboard due to disabilities, injuries, or other impairments. The use of facial movement recognition to control the mouse cursor could provide a hands-free way to interact with computers, thereby increasing accessibility. And the objective is to create a program using a computer's camera to collect real-time video containing human faces. Recognize human faces using the dlib library and the trained model 'shape_predictor_68_face_landmarks'. From this, perform operations to move the mouse cursor on the screen corresponding to the facial movements.

TABLE OF CONTENT

CHAPTER 1: THEORETICAL BASIS OF IMAGE PROCESSING

1. Introduction to Image Processing

Image processing is a field of science and technology. It is a relatively new scientific field compared to many others, but its development pace is very fast, stimulating research centers and applications, especially computers dedicated to it.

Image processing methods start with the main applications: improving image quality and image analysis. The first known application was to enhance the quality of newspaper images transmitted by cable from London to New York in the 1920s. The issue of improving image quality is related to the distribution of brightness levels and the resolution of the image. Image quality improvement was developed around 1955. This can be explained by the fact that after World War II, computers developed rapidly, facilitating the process of digital image processing. In 1964, computers were capable of processing and improving the quality of images from the moon and the United States' Ranger 7 satellite, including: highlighting edges, storing images. From 1964 to the present, the means of processing, improving quality, and recognizing images have been continuously developed. Artificial intelligence methods such as artificial neural networks, modern processing algorithms and improvements, and image compression tools are increasingly widely applied and yield many promising results.

Image processing plays an important role in the interaction between humans and computers. The image recognition processing is a process that includes operations to transform an input image to produce a result or a conclusion.

First, natural images from the outside world are captured through devices (such as cameras, photography cameras). Then, through image processing, the image is directly converted into a digital image, which facilitates further processing. The results of image processing can be: producing a better image according to the user's desire; analyzing images to obtain information for image classification and recognition; drawing observations, conclusions, etc.

1.1 The basic steps in image processing process

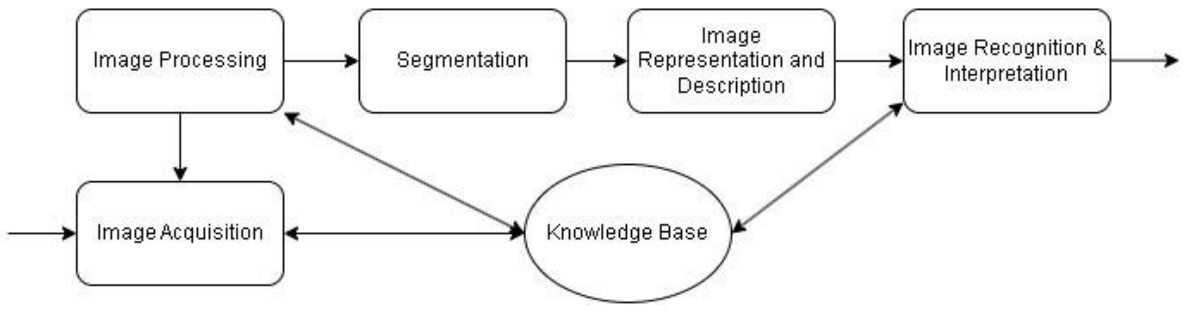


Image Processing
Segmentation
Image Representation and Description
Image Recognition & Interpretation
Image Acquisition
Knowledge Base

1.1.1 Image Acquisition

Images can be captured through color or black and white cameras, scanners, camcorders, etc. The quality of a captured image depends on the capturing device and the environment (lighting, scenery). Subsequently, the image is converted through ADC (image digitization). The ADC (Analog to Digital Converter) process is used to acquire the digital form of the image.

1.1.2 Image Processing

After capture, the image might have low contrast noise, so it needs to go through a pre-processing unit to enhance its quality. The main function of the image pre-processing unit is noise filtering and contrast enhancement to make the image clearer and sharper. The image will be improved in terms of contrast, noise reduction, image restoration, geometric adjustment, etc.

- **Noise reduction**: there are two types of noise: systematic noise and random noise. The characteristic of systematic noise is its periodicity, so it can be removed by using the Fourier transform and eliminating the peaks. Random noise can be reduced by interpolation methods, median filtering, and average filtering.
- **Gray level adjustment**: is to correct the non-uniformity of the capturing device or the contrast between different areas of the image.
- **Image scatter correction**: Images obtained from optical or electronic devices can be blurred or smeared. The Fourier transform method is based on the convolution of the image with a scatter function.

1.1.3 Segmentation

Image segmentation is the process of dividing an input image into its component regions for representation, analysis, and recognition of the image. For example, to recognize text (or barcodes) on an envelope for the purpose of sorting mail, it is necessary to divide the sentences, words about the address or the person's name into separate words, letters, numbers (or bars) for recognition. This is the most complex and difficult part of image processing and is also prone to errors, potentially reducing the accuracy of the image. The result of image recognition depends greatly on this process. The result of image segmentation is usually raw pixel data, functions containing the edges of an image region, or the set of all pixels within that region.

1.1.4 Image Representation and Description

Image representation: The output image after segmentation contains the pixels of the image region (segmented image) along with codes linked to adjacent regions. Transforming this data into a suitable form is necessary for further processing by computers. The selection of properties to represent an image is called feature selection, which involves separating the characteristics of the image in the form of quantitative information or as a basis for distinguishing one class of objects from another within the received image scope.

Image description: After being digitized, the image will be stored in memory or transferred to the next stages for image analysis. Storing images directly from raw images requires a very large memory capacity and is not efficient for subsequent applications. Typically, these raw images are represented or encoded according to the characteristics of the image, called features

such as image edges, image regions. Some methods of image representation include:
- Run-Length Encoding (RLE): This method is often used for representing image regions and is applied to binary images. An image region R can be simply encoded using a binary matrix:

- $U(m, n) = 1$, nếu $(m, n) \in R$
- $U(m, n) = 0$, nếu (m, n) không $\in R$

Where U(m, n) is a function describing the gray level of the image at coordinates (m, n). With the representation above, an image region is described by a set of strings of 0s or 1s. For instance, if we describe a binary image of a region represented by coordinates (x, y) along the axes and specify only for the value "1", then the description could be: (x, y) r; where (x, y) are the coordinates, and r is the number of consecutive bits with the value "1" horizontally or vertically.

- Chain Code: This method is typically used to represent the borders of an image. Any arbitrary image border is divided into small segments. Connecting these points, we get successive straight segments that are assigned a direction, creating a chain consisting of segments. The directions can be chosen as 4, 8, 12, 24, etc., and each direction is encoded in decimal or binary to form the code for the direction.

- Quadtree Encoding: This is commonly used to encode image regions. The initial image region is divided into four often equal parts. If each region is homogeneous (containing all black (1) or white (0) pixels), that region is assigned a code and is not divided further. Non-homogeneous regions are further divided into four parts following the procedure above until all regions are homogeneous. The codes for the subdivisions create a tree that represents the division into homogeneous regions.

1.1.5 Image Recognition & Interpretation

Image recognition is the process of identifying and classifying objects or features within an image, and often involves comparing the image with a set of known patterns or templates. Interpolation, in this context, refers to the process of making sense of the recognized image by inferring additional information. For instance, recognizing a series of digits and dashes on an envelope could lead to the interpolation of a phone number.

Image recognition can be theoretically categorized into two basic types:

Parametric Recognition: This approach involves recognizing images based on parameterized models of the image content. In parametric recognition, the system uses a set of quantifiable aspects of the image, such as statistical properties or geometric parameters, to identify and classify the object.

Structural Recognition: Structural recognition relies on the understanding of the image's structure. This may include the relationships and arrangements between different parts of an image. It often uses topological and syntactical information to recognize patterns and objects.

Common applications of image recognition technologies in science and technology include:

- Character Recognition: This includes the recognition of printed text, handwritten text, and electronic signatures. Optical Character Recognition (OCR) systems are a common example, where printed or handwritten text is converted into machine-encoded text.
- Text Recognition: Beyond individual characters, this involves recognizing and interpreting blocks of text, which can be used in document analysis, natural language processing, and information retrieval.
- Fingerprint Recognition: This biometric method identifies individuals based on the unique patterns found in their fingerprints. It is widely used for security and identification purposes.
- Barcode Recognition: Barcodes are optical machine-readable representations of data, which can be found on various products for identification. Barcode recognition systems decode the information contained in the barcode.
- Facial Recognition: This technology identifies or verifies a person from a digital image or video frame. It has applications in security, law enforcement, and user authentication.

1.1.6 Knowledge Base

An image is a rich but complex source of data due to its characteristics in terms of lines, light and dark areas, pixel resolution, and noise from the image capture environment. In image processing and analysis, the goal is not only to simplify mathematical methods for convenience in processing but also to mimic the process of receiving and processing visual information like humans do. To simulate human cognitive abilities, many modern techniques in image processing have been developed based on the principles of human intelligence, where knowledge and experience are used to improve the processing.

The steps in digital image processing may vary depending on the specific application requirements:

- **After digitization**: The image is usually compressed to reduce storage requirements and optimize transmission.
- **Quality improvement**: If the digitized image does not meet the quality requirements, steps such as smoothing, noise removal, and contrast enhancement can be applied.
- **Skipping steps**: For images that meet quality standards, it's possible to skip the quality improvement step and move directly to image segmentation or feature extraction.

Feature extraction is an important step in the object recognition process on images. Effective selection of image characteristics helps make recognition more accurate and efficient in terms of computation time and storage capacity reduction. Some key features in images include:

- **Spatial features**: These include the distribution of gray levels, the probability of gray level occurrences, the amplitude of features, inflection points, etc.
- **Transformation features**: These features are often extracted through zonal filtering, using feature masks with various shapes such as rectangles, squares, triangles, and others.
- **Edge and boundary features**: This characteristic is related to clearly determining the boundaries of objects in images and extracting invariant attributes that can be used in object recognition. Methods such as the Laplacian operator, gradient operator, compass

operator, and zero crossing are all used to detect and extract information about edges and boundaries.

The methods above support the extraction of important information from images, thereby improving the accuracy of recognition and classification processes. Image processing technology continues to evolve with the support of other fields such as machine learning and artificial intelligence, providing the ability to simulate visual perception and cognition similar to humans, thus expanding its application possibilities in various industries.

1.2 Some basic issues in image processing

1.2.1 Picture Element

The origin of an image (natural image) is continuous in space and brightness. For computer processing, the image needs to be digitized. Image digitization is the approximate transformation of a continuous image into a set of points that match the real image in terms of location (space) and brightness (gray scale). The distance between these pixels is set so that the human eye cannot distinguish the boundaries between them. Each such point is called a pixel (PEL: Picture Element). In the context of two-dimensional images, each pixel corresponds to a coordinate pair (x, y).

Definition: A pixel is an element of a digital image at coordinates (x, y) with a specific gray scale or color. The size and distances between these pixels are chosen appropriately so that the human eye perceives the continuity of space and gray scale (or color) of the digital image as nearly identical to the real image. Each element in the matrix is called an image element.

1.2.2 Resolution

The resolution of an image is the pixel density specified on a displayed digital image. The distance between pixels must be chosen so that the human eye can still perceive the continuity of the image. Selecting the appropriate distance creates a resolution density and is distributed along the x and y axes in two-dimensional space.

1.2.3 Gray level of the image

A pixel has two basic characteristics: the position (x, y) of the pixel and its gray level. The gray level of a pixel is the intensity of its brightness assigned by a numerical value at that point. Common gray scale value ranges include: 16, 32, 64, 128, 256 (with 256 being the most commonly used level. This is because computer technology uses 1 byte (8 bits) to represent the gray level: $2^8 = 256$ levels, meaning from 0 to 255).

A black and white image (grayscale image): is an image that has only two colors, black and white (and contains no other colors), with the gray level at different pixels potentially varying.

Binary image: an image that has only 2 distinct black and white levels, i.e., 1 bit describes 2^1 different levels. In other words, each pixel of a binary image can only be 0 or 1.

Color image: within the framework of the three-color theory (Red, Blue, Green) used to create the world of color, 3 bytes are typically used to describe the color level, thus the color values are: $2^{8*3} = 2^{24} \approx 16.7$ million colors.

1.2.4 Definition of a digital image

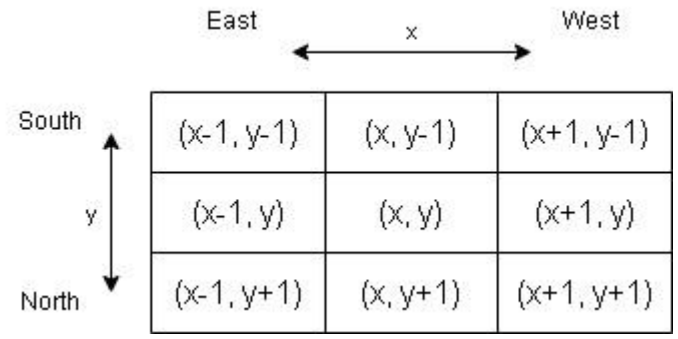
A digital image is a collection of pixels with appropriate gray levels used to describe an image that closely resembles the real image.

1.2.5 Relationship between pixels

An image, let's assume, is represented by the function f(x, y). The subset of pixels is S, and a pair of related pixels is denoted as p, q.

Pixel Neighbors: Suppose there is a pixel p at coordinates (x, y), p has 4 nearest neighbors in the vertical and horizontal directions (which can be considered as neighbors in the 4 main directions: East, West, South, North).

where: the number 1 represents a logical value; N4(p) is the set of 4 neighboring pixels of p.



Diagonal Neighbors: The diagonal neighbors NP(p) (which can be considered as diagonal neighbors in the 4 directions: Southeast, Northeast, Southwest, Northwest)

$N_P(p)$ = { (x+1, y+1); (x+1, y-1); (x-1, y+1); (x-1, y-1)}

Combined Set: $N_8(p) = N_4(p) + N_P(p)$ is the set of 8 neighboring pixels of p. Note: If (x, y) is at the edge of the image, some points will be outside the image.

Pixel Linkages: Linkages are used to determine the boundaries of objects or regions within an image. A linkage is characterized by the adjacency between points and their gray levels. Suppose V is the set of gray level values. An image with brightness values on a gray scale from 32 to 64 is described as follows: V={ 32, 33, …, 63, 64}. There are 3 types of linkages:

- 4-linkage: Two pixels p and q are said to be 4-linked with brightness values V if q is within one of the neighbors of p, that is, q belongs to $N_4(p)$.
- 8-linkage: Two pixels p and q are within one of the 8 neighbors of p, that is, q belongs to $N_8(p)$.
- m-linkage (mixed linkage): Two pixels p and q with brightness values V are said to be m-linked if: q is within $N_4(p)$ or q is within $N_P(p)$.

- Measuring the distance between pixels:

Definition: The distance D(p, q) between two pixels p with coordinates (x, y) and q with coordinates (s, t) is a distance function or Metric if:

1. $D(p, q) \geq 0$ (with $D(p, q) = 0$ if and only if $p = q$).

2. $D(p, q) = D(q, p)$.

3. $D(p, z) \leq D(p, q) + D(q, z)$; where z is another pixel.

Euclidean Distance: The Euclidean distance between two pixels p(x, y) and q(s, t) is defined as follows:

$$D_e(p, q) = [(x - s)^2 + (y - t)^2]^{1/2}$$

City Block Distance: The distance D4(p, q) is called the City Block Distance (or Manhattan Distance) and is defined as follows:

$$D_4(p, q) = |x - s| + |y - t|$$

Radius Value between Pixels r: The radius value r between a pixel from the center of pixel p to the center of another pixel q.

The distance $D_8(p, q)$, also known as the Chessboard Distance, between pixels p and q is defined as follows:

$$D8(p, q) = \max(|x - s|, |y - t|)$$

1.2.6 Image Transform

In image processing, due to the large number of pixels and extensive calculations (high computational complexity), a large memory capacity and long computation times are required. Classical scientific methods applied to image processing are mostly unfeasible. Equivalent operations are used or transformations into different processing domains are made to simplify the calculations. After the easier processing has been performed, inverse transformations are used to return to the original domain. Common transformations encountered in image processing include:

- Fourier, Cosine, Sine transformations...
- Image transformations (representations) by convolution, Kronecker products (according to signal processing). Other transformations like KL (Karhunen-Loève), Hadamard.
- Some statistical probability tools are also used in image processing.

1.2.7 Image Compression

Regardless of their format, images still occupy a considerable amount of memory space. To address this, image compression techniques have been introduced. The stages of image compression can be divided into first-generation and second-generation methods. Currently, MPEG standards are being effectively utilized with images.

2. Sampling and quantization

An image g(x, y) recorded by a camera is a continuous two-dimensional plane image. This image needs to be converted into a suitable form for processing by a computer. The method of transforming an image (or a function) from a continuous form in space and value into a discrete numerical form is called image digitization. This transformation can include two steps:

Step 1: Measuring values over spatial intervals called sampling.

Step 2: Mapping the measured intensity (or value) to a finite number of discrete levels called quantization.

2.1 Sampling

Sampling is a process through which an image created over a continuous area is converted into discrete values according to integer coordinates. This process involves two choices: One is the sampling interval, and the other is the representation of the sample form. The first choice is ensured by Shannon's sampling theorem. The second choice relates to the Metric used in the discrete domain.

- Sampling Interval:

The sampled image can be described as the selection of a set of sampling positions in a continuous two-dimensional space. First, it is described through the one-dimensional sampling process using the delta function:

$$\delta(x - x_0) = \begin{cases} 0, & khi\ x \neq 0 \\ \infty, & khi\ x = 0 \end{cases}$$

$$\int_{-\infty}^{\infty} \delta(x - x_0)dx = \int_{x_0-}^{x_0+} \delta(x - x_0)dx = 1$$

The comb function is a series of comb-like impulses from (-∞, +∞) with intervals Δx:

$$Comb(x) = \sum_{r=-\infty}^{\infty} \delta(x - r\Delta x)$$

with r being an integer and Δx is the sampling interval.

The sampling interval Δx is a parameter that must be chosen sufficiently small and appropriate; otherwise, the true signal cannot be recovered from sampled signal.

- Shannon Sampling Theorem:

Assume g(x) is a band-limited function and its Fourier transform is $G(\omega x) = 0$ for all values $W_x > \omega_x$. Then g(x) can be completely recovered from its samples taken at regular intervals Δ, that is $\Delta x \leq \frac{1}{2\omega_x}$

Shannon's sampling theorem can be extended to two-dimensional space. The two-dimensional comb function is then defined as:

$$Comb(x,y) = \sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty} \delta(x - i\Delta x)(y - j\,\Delta y)$$

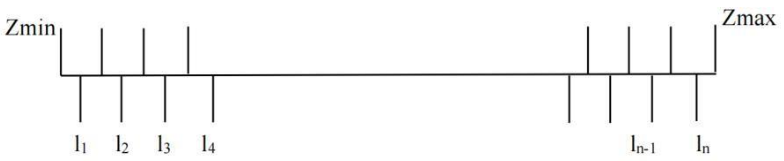
Khi đó: $\Delta x \leq \frac{1}{2\omega_x}; \Delta y \leq \frac{1}{2\omega_y}$

- Tesselation:

The pixel sampling pattern refers to the arrangement of pixels in two-dimensional space. Some pixel sampling patterns include rectangular, triangular, hexagonal, etc.

2.2 Quantization

Quantization is the mapping from the real numbers that describe the sampled values to a finite range of real numbers. In other words, it is the process of digitizing the amplitude.



The sampled values Z are a set of real numbers ranging from the minimum value Z_{min} to the maximum Z_{max}. Each number within the set of sampled values Z needs to be transformed into a finite set of bits for the computer to store or process.

Assume Z is a sampled value (a real number) at some position on the image plane, and $Z_{min} <= Z' <= Z_{max}$, and suppose we want to quantize that value into one of the discrete levels: l1, l2, …, ln corresponding to Z_{min} to Z_{max}. Then, the quantization process can be performed by dividing the entire domain (Z_{max} - Z_{min}) into l intervals, each interval being Δl, and the I-th interval centered at the midpoint of the adjacent intervals l_i. The values of z are quantized and represented by li according to the above process; the error of the quantization process can be determined by: $e_q = l_i - Z$.

CHAPTER 2: FACE RECOGNITION

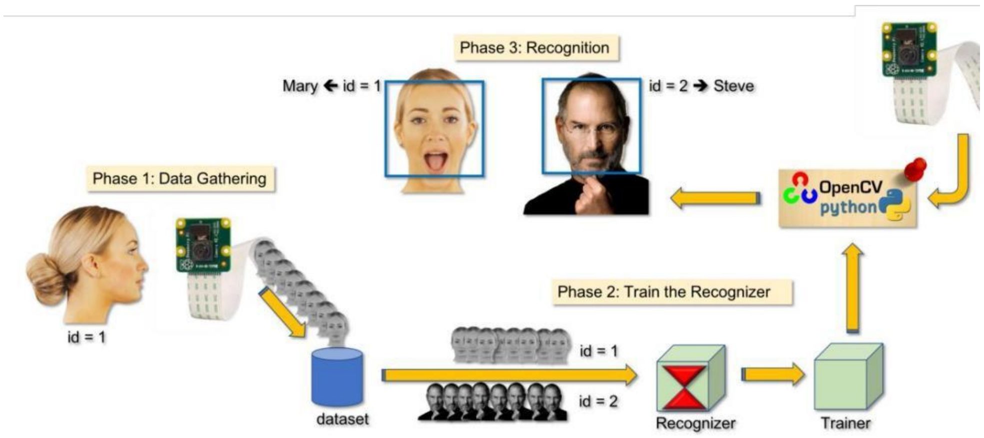
1. Face recognition model

1.1 Facial recognition

Facial recognition is a technique for identifying or verifying faces from digital images or video frames. A human can quickly recognize faces without much effort. It is an easy task for us, but it is a challenging task for computers. There are various complexities involved, such as low resolution, occlusion, changes in lighting, etc. These factors significantly affect the accuracy of computers in effectively recognizing faces. First, it's important to understand the difference between face detection and face recognition.

Face Detection: Face detection is usually considered the process of locating faces (position and size) within an image and potentially extracting them for use by face recognition algorithms.

Face Recognition: Face recognition algorithms are used to find unique descriptive features in



General Steps:

Step 1: Collect facial data (Face Data Gathering).

Step 2: Train the model with the collected data (Train the Recognizer).

Step 3: Recognize and differentiate between faces (Recognition).

=> In step 1 (collecting facial data) and step 3 (recognizing faces), it is crucial to determine the location of the face.

1.2 Face Detecting

The important task and foundation for face recognition (Face Recognition) is Face Detection (Face Detecting). First and foremost, the face must be "captured" (in Step 1) in order to recognize and differentiate it from other faces (in Step 3). And the most common method currently used for face detection is employing the Haar Cascade classifier.The important task and foundation for face recognition (Face Recognition) is Face Detection (Face Detecting). First and foremost, the face must be "captured" (in Step 1) in order to recognize and differentiate it from other faces (in Step 3). And the most common method currently used for face detection is employing the Haar Cascade classifier.

1.3 Introduction to Haar Cascade

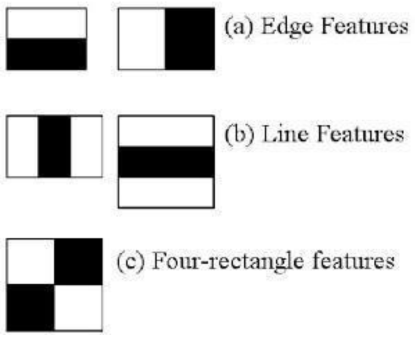
Haar Cascade is an algorithm that was created based on those features to detect objects (which could be faces, eyes, hands, items, etc.) proposed in 2001 by Paul Viola and Michael Jones in their paper with the assertion "Rapid Object Detection using a Boosted Cascade of Simple Features".

The initial implementation was used to detect frontal faces and features like Eyes, Nose, and Mouth. However, there are many pre-trained Haar features available on GitHub that can also be used for other objects as well as for the entire body, upper body, lower body, smiles, and many other items.

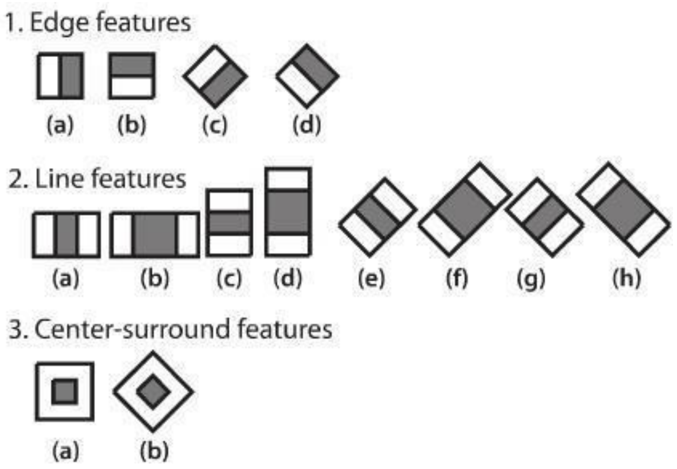
To put it more simply, what is a Haar Cascade? It is a classifier that can help us in facial recognition (Haar Cascade face detection). Haar Cascade employs layers of Haar features and then uses many of these features in multiple stages (Cascades) to create a complete face recognition machine.

1.4 Haar Features (Haar Filters)

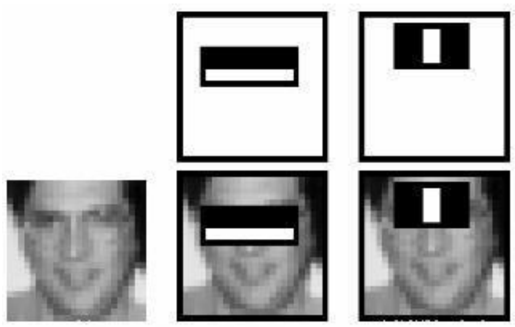
Examples of Haar features are listed below, where a) captures the edges in an image, and b) captures the lines in the image. Additionally, there are other Haar features, such as the "four-rectangle" feature example c) shown below.



or features that are located within the center of a region, as in example 3 in the image below:



However, the application of these filters is slightly different compared to the convolutional windows in CNN. In CNN, the filter spans the entire sliding window, while in Haar features, the filter only occupies a part of the sliding window. This is illustrated in the following image:



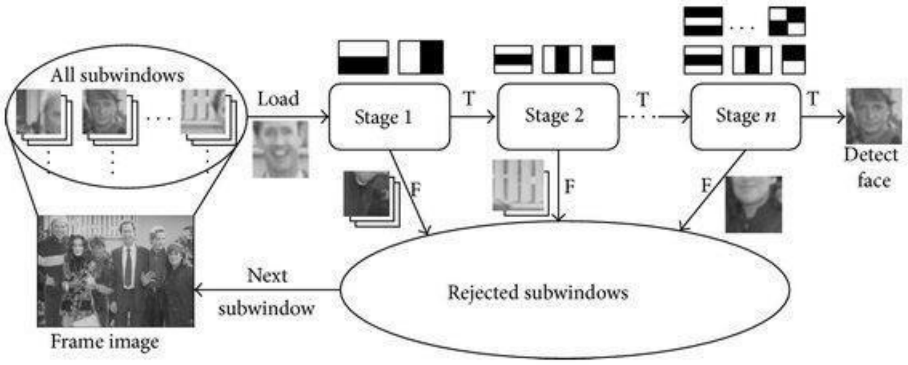
In the image above, the sliding window is neatly positioned to encompass the entire image. It can be observed that the first filter is looking for an "edge" that separates the eyes/brows from the nose, due to the significant color difference in this area;

In the subsequent filter, the model is searching for the bridge of the nose, as it will be lighter in color compared to the sides (because it is raised and catches the light more easily). And as mentioned before, the Haar filter specifically looks at a region within the window to find features: in a face, the nose is always in the center, not at the edges, so there is no need to look at the other corners.

The layered structure of Haar Cascade

To further understand how Haar Cascade works, we can delve deeper into the steps of face recognition by Haar Cascade:

Among the 6000+ features, they are divided into many stages. Each time a sliding window passes over a region of the image, each step is processed: if step 1 identifies it as a face, we move to step 2; if not, we skip that region and slide the window to another area. If a region passes all the stages of face tests, then that window contains a human face.



Step 1: The image (that has been sent to the classifier) is divided into small sections (or sub-windows as illustrated in the figure).

Step 2: We place N detectors in a cascading manner where each detector recognizes a combination of different types of features from the images (for example: lines, edges, circles, squares) that are passed through. Assume that when the extraction of geographical objects is performed, each sub-section is assigned a confidence value.

Step 3: The image (or sub-image) with the highest confidence level is detected as a face and is sent to the accumulator, while the rest are rejected. Thus, the Cascade fetches the next frame/image if any remain and starts the process over.

1.5. Face Feature Execution

At this step, we need a model to extract features from the face image (Feature Extractor). The features extracted by this model are also referred to as Face Embeddings or Face Encodings, which can be understood as a different representation of the face image that facilitates easier face recognition (instead of using raw pixel values). Face Embeddings have a very interesting property: similar faces will have "closer" Face Embeddings (based on Euclidean distance).

2. Challenges in Facial Recognition

Face recognition systems are essential nowadays and have come a long way. Their usage is necessary in some applications, such as image retrieval systems, surveillance, authentication/access control, etc. However, there are several ongoing challenges in image or face recognition systems.

These challenges need to be overcome to create more effective face recognition systems. Here are the challenges that affect the capabilities of Face Recognition Systems.

Lighting

Lighting plays a crucial role in the image recognition process. If there is a slight change in lighting conditions, it can greatly affect the outcome. That's because as lighting changes, the results may differ for the same subject causing low or high light.

Background

The background of the subject also plays an important role in Face Recognition. The results may not be the same outdoors as what is produced indoors because factors – affecting its performance change immediately after the location changes.

Pose

The face recognition system is very sensitive to pose variations. Head movements or different positions of the camera can cause changes in the facial structure and it will produce incorrect results.

Variability

Variability means things that are often on the face like beards, mustaches, accessories (safety glasses, baseball caps, masks, etc.) also affect the estimates of the face recognition system.

Expressions

Another important factor to keep in mind is the different expressions of the same individual. Changes in facial expressions can produce a different result for the same person.

CHAPTER 3: INTRODUCTION TO FACIAL LANDMARKS AND DLIBS

1. Introduction to Facial Landmarks

Determining facial landmarks is a sub-problem of the shape prediction problem. So, what is shape prediction? It involves identifying the key points that define the shape of an object in an image. In the task of determining facial landmarks, we need to identify the key points in the image that form the shape of a human face. Facial landmarks are inputs for many other problems such as head pose estimation, face swapping, blink detection, face alignment, and notably, the FaceID recognition technology equipped by Apple on the iPhone X and later models.

The general process for detecting facial landmarks includes two steps:

Step 1 - Detect Face: Locate the position of the face in the image (detect face ROI). Some methods include Haar cascades, HOG + Linear SVM object detector, pre-trained DL-based models, etc. Regardless of the method, eventually, we get a face bounding box (coordinates).

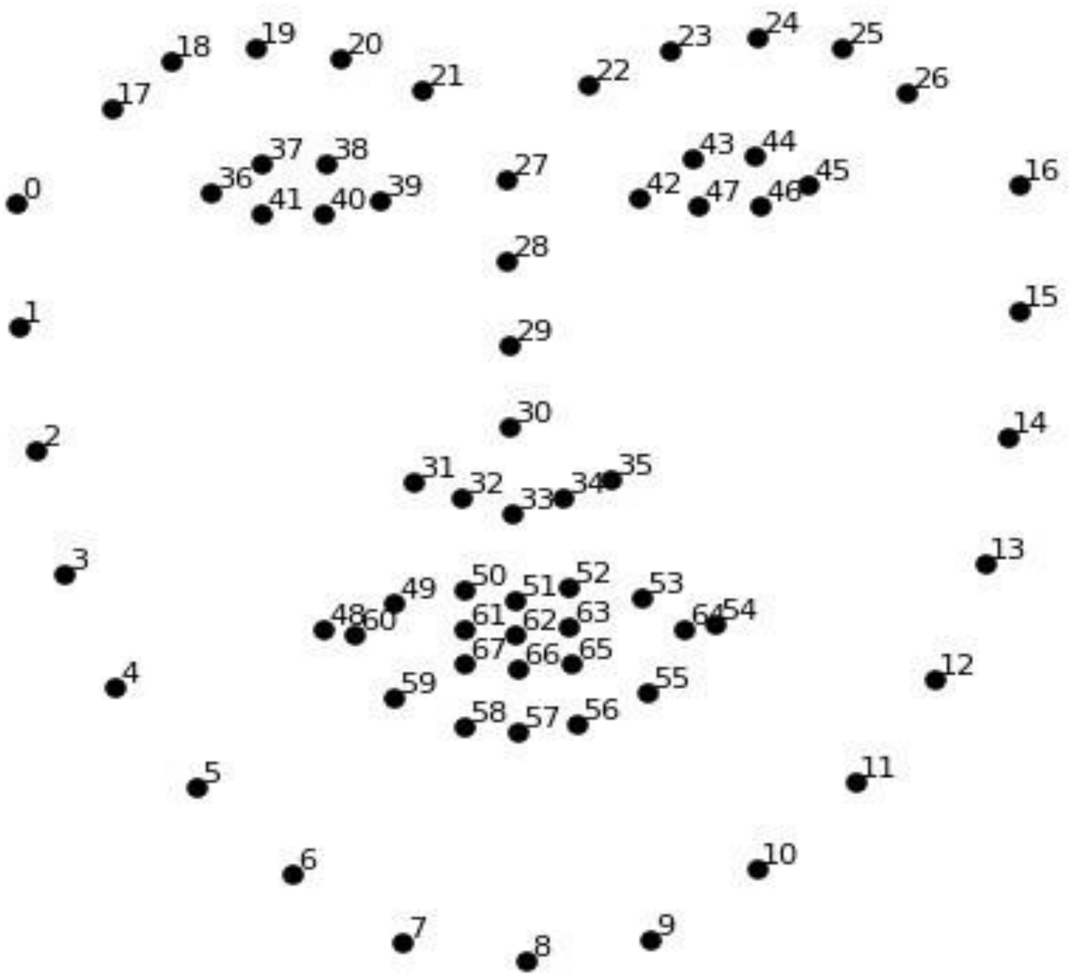
Step 2 - Detect Key Facial Structures: On the face ROI (found in step 1), detect key facial structures. There are many facial landmark detectors, but all methods try to localize and label the facial regions:

- Mouth
- Right eyebrow
- Left eyebrow
- Right eye
- Left eye
- Nose
- Jaw

2. Exploring Dlib’s Facial Landmark Detector

The pre-trained facial landmark detector in Dlib is used to estimate the 68 (x, y) coordinates that correspond to the facial landmarks on the face.

The indices of the 68 coordinates can be represented as shown in the figure below (however, in Python, they will be numbered from 0 to 67).



From there, we can see the coordinates of the regions on the face as follows:

- Mouth: [49;68]
- Right eyebrow: [18;22]
- Left eyebrow: [23;27]
- Right eye: [37;42]
- Left eye: [43;48]
- Nose: [28;36]
- Jaw: [1;17]

CHAPTER 4: DESIGN AND IMPLEMENTATION OF THE PROJECT

1. Methods of implementation

After detecting the subject's face and identifying 68 reference points on the front part of the face, we sequentially determine the movements on the face based on the distance between the points. Then we perform the operations of moving, clicking, and speeding up the movement of the mouse cursor:

For the eye area, we choose the distance between points 28 and 29 as the reference value because the distance between these two points changes little in proportion to the face when moving the face closer or farther and in different directions. Since the training set can only return the result of both eyes being closed or open at the same time, we only need to determine when the left (or right) eye closes, by calculating the ratio of the distance between points 38 - 40 compared to the reference distance. When the eye closes, perform a click of the mouse cursor on the screen.

For the mouth area, we choose the distance between points 62 - 66 to determine when the mouth is opened. When the mouth opens, we will increase the mouse movement speed (default 20, max 80, step 20).

For facial movements, we compare the ratio of the distance between the sides of the face for leaning left (13 - 30 / 30 - 2) and leaning right (2 - 30 / 30 - 13). For looking up, we also perform comparisons of the distance on the points along the face (8 - 33 / 33 - 27). For looking down, we compare the distance between the width of the open eyes and the distance from the eye to the eyebrow (23 - 43 / 43 - 47), because when we look down our eyes tend to open wider. The directions of turning up, down, left, right correspond to the operations of moving the mouse up, down, left, and right on the screen.

The results are displayed directly in real-time on the computer screen. After that, we can fine-tune to return more reasonable results and smoother interaction for the user.

2. Summary libraries and source code

A. Libraries:

- OpenCV: Used to read images from a camera, preprocess images, overlay text on frames, and display results.

- Numpy: Used for performing mathematical calculations.

- Dlib: Utilized to obtain the pre-trained facial recognition model for the frontal face, in conjunction with the shape_predictor_68_face_landmarks dataset.

- Time: Time.sleep allows for pausing the process for a set duration to avoid uncontrolled repeated actions.

- PyAutoGUI: Executes controls related to the mouse cursor on the screen.

B. Source code:

```
import cv2
import numpy as np
import dlib
from time import sleep
import pyautogui

# Setup
cursor_speed = 20  # Cursor speed
loop_time_count = 0  # Count the actual time of the loops

# File destination
direct = 'D:\\Downloads\\Ton Minh\\'
datfile = direct + 'shape_predictor_68_face_landmarks.dat'

# Initialize the variable to read images from the camera
cap = cv2.VideoCapture(0)

# Initialize functions to detect and predict
face_detector = dlib.get_frontal_face_detector()
shape_predictor = dlib.shape_predictor(datfile)

# Function to calculate the Euclidean distance between two points
def calculate_distance(point1, point2):
    return np.sqrt((point1[0] - point2[0]) ** 2 + (point1[1] - point2[1]) ** 2)

# Function to draw facial landmarks on the frame
def draw_facial_landmarks(landmarks, frame):
    for i in range(68):
        coordinates = (landmarks.part(i).x, landmarks.part(i).y)
        cv2.circle(frame, coordinates, 0, (255, 0, 0), 5)

# Function to extract the coordinate values of a landmark point
def get_landmark_point(landmarks, index):
    return (landmarks.part(index).x, landmarks.part(index).y)

# Function to handle eye aspect ratio and blinking
def handle_eye_aspect_ratio(landmarks, frame):
    left_eye_width = calculate_distance(get_landmark_point(landmarks, 38),
get_landmark_point(landmarks, 40))
    right_eye_width = calculate_distance(get_landmark_point(landmarks, 43),
get_landmark_point(landmarks, 47))
```

```
    base_distance = calculate_distance(get_landmark_point(landmarks, 28), get_landmark_point(landmarks,
29))

    # Detect if eyes are closed
    if left_eye_width < (base_distance * 7.5 / 18) or right_eye_width < (base_distance * 7.5 / 18):
        cv2.putText(frame, 'Eyes closed, click!', (230, 20), cv2.FONT_HERSHEY_DUPLEX, 0.5, (255, 0, 0))
        pyautogui.click()  # Mouse click
    else:
        cv2.putText(frame, 'Eyes opened', (230, 20), cv2.FONT_HERSHEY_DUPLEX, 0.5, (0, 255, 0))

# Function to handle mouth movement
def handle_mouth_movement(landmarks, frame, current_speed):
    mouth_opening = calculate_distance(get_landmark_point(landmarks, 62), get_landmark_point(landmarks,
66))
    cv2.putText(frame, 'M: ' + str(round(mouth_opening, 2)), (10, 130), cv2.FONT_HERSHEY_DUPLEX, 0.5,
(255, 0, 0))
    if mouth_opening > 8:
        cv2.putText(frame, 'Talking', (255, 50), cv2.FONT_HERSHEY_DUPLEX, 0.5, (0, 0, 255))
        # Throttle the cursor speed changes and add a small delay to prevent rapid changes
        sleep(0.2)
        new_speed = min(80, current_speed + 20)
    else:
        new_speed = 20

    return new_speed

# Function to detect face orientation and move the cursor accordingly
def handle_face_orientation(landmarks, frame, current_speed):
    ratio_right = calculate_distance(get_landmark_point(landmarks, 2), get_landmark_point(landmarks,
30)) / \
                  calculate_distance(get_landmark_point(landmarks, 30), get_landmark_point(landmarks,
13))
    ratio_left = calculate_distance(get_landmark_point(landmarks, 13), get_landmark_point(landmarks,
30)) / \
                 calculate_distance(get_landmark_point(landmarks, 30), get_landmark_point(landmarks,
2))

    if ratio_right >= 1.5:
        cv2.putText(frame, 'Turn Right', (355, 110), cv2.FONT_HERSHEY_DUPLEX, 0.5, (0, 0, 255))
        pyautogui.move(current_speed, 0, duration=pyautogui.MINIMUM_DURATION)
    elif ratio_left >= 1.5:
        cv2.putText(frame, 'Turn Left', (155, 110), cv2.FONT_HERSHEY_DUPLEX, 0.5, (0, 0, 255))
        pyautogui.move(-current_speed, 0, duration=pyautogui.MINIMUM_DURATION)
    else:
        ratio_up = calculate_distance(get_landmark_point(landmarks, 8), get_landmark_point(landmarks,
33)) / \
                   calculate_distance(get_landmark_point(landmarks, 33), get_landmark_point(landmarks,
27))
        ratio_down = calculate_distance(get_landmark_point(landmarks, 23),
get_landmark_point(landmarks, 43)) / \
                     calculate_distance(get_landmark_point(landmarks, 43),
get_landmark_point(landmarks, 47))



        print("Outside: " + "Ratio up: " + str(ratio_up) + " - Ratio down: " + str(ratio_down) + "\n")
        if ratio_down < 3 and ratio_up < 1.5 :
            print("Look down" + "Ratio up: " + str(ratio_up) + " - Ratio down: " + str(ratio_down) +
"\n")
            cv2.putText(frame, 'Turn Down', (255, 390), cv2.FONT_HERSHEY_DUPLEX, 0.5, (0, 0, 255))
            pyautogui.move(0, current_speed, duration=pyautogui.MINIMUM_DURATION)
        elif ratio_up >= 1.5:
            print("Look up" + "Ratio up: " + str(ratio_up) + " - Ratio down: " + str(ratio_down) +
"\n")
            cv2.putText(frame, 'Turn Up', (255, 90), cv2.FONT_HERSHEY_DUPLEX, 0.5, (0, 0, 255))
            pyautogui.move(0, -current_speed, duration=pyautogui.MINIMUM_DURATION)
        else:
            print("No ")
            print("Normal: " + "Ratio up: " + str(ratio_up) + " - Ratio down: " + str(ratio_down) +
"\n")
            cv2.putText(frame, 'Straight look', (255, 110), cv2.FONT_HERSHEY_DUPLEX, 0.5, (0, 0, 255))



while True:
    loop_time_count += 1
    ret, frame = cap.read()
    if not ret:
        break
    frame = cv2.flip(frame, 1)
    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    detected_faces = face_detector(frame)
    for face in detected_faces:
        facial_landmarks = shape_predictor(gray_frame, face)

        draw_facial_landmarks(facial_landmarks, frame)
        handle_eye_aspect_ratio(facial_landmarks, frame)
        cursor_speed = handle_mouth_movement(facial_landmarks, frame, cursor_speed)
        handle_face_orientation(facial_landmarks, frame, cursor_speed)

        # Display time
        cv2.putText(frame, 'Time: ' + str(round(loop_time_count / 30, 2)), (10, 210),
cv2.FONT_HERSHEY_DUPLEX, 0.5, (255, 0, 0))

    # Display camera feed
    cv2.imshow('App', frame)
    if cv2.waitKey(1) == 27:  # 27 is the ESC key
        break
```

3. Results

The video of the implementation results has been uploaded by us at the following public drive link:

Full screen video:

https://drive.google.com/file/d/1vhwfcGkaV8o_DeT7QUe8j15mfZ5E0--L/view?usp=sharing

Source code:

https://github.com/tonminhce/machine-learning-project

CHAPTER 5: CONCLUSION

The topic does not dive deeply into facial recognition algorithms and data training but stops at using pre-existing models as well as performing simple calculations on the returned training values. Nevertheless, learning about the face-landmarks model has given us a deeper understanding of facial recognition and working with the output data has also helped us envision how an AI application could be implemented in reality.

The results achieved meet the set requirements, but there are still many areas that need to be optimized, as well as the development of more reasonable and logical algorithms. The project could be further developed to add more features such as drag and drop, scrolling, or simply increasing efficiency (mouse speed, movement accuracy, faster mouse movement with more head tilt, etc.).