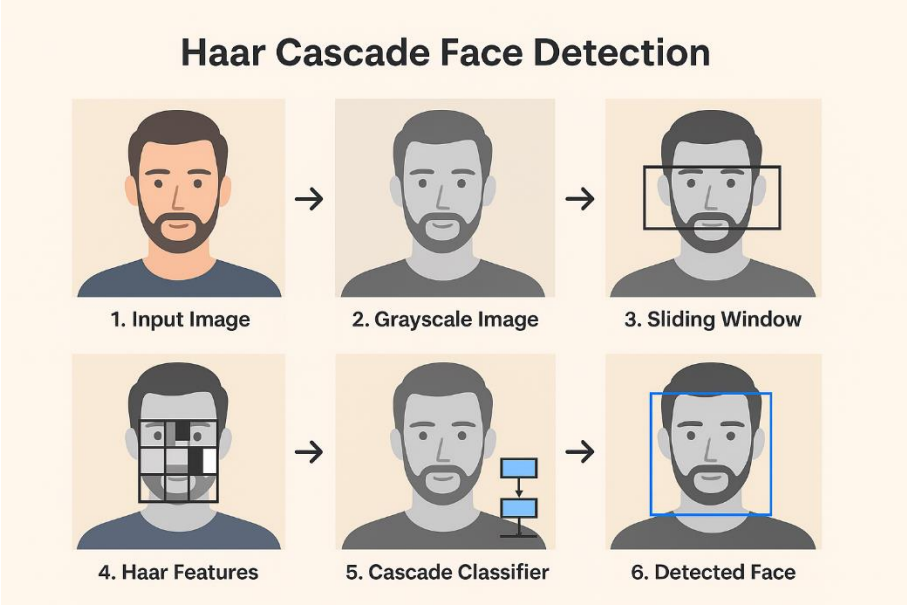
Modul Praktikum Pengenalan Wajah dengan Algoritma Haar Cascade

1. Apa Itu Haar Cascade?

Haar Cascade adalah algoritma deteksi objek berbasis fitur, diperkenalkan oleh Viola dan Jones (2001) dalam paper legendaris "Rapid Object Detection using a Boosted Cascade of Simple Features".



Tujuannya: mendeteksi objek seperti wajah, mata, atau tubuh manusia secara cepat dalam gambar atau video.

2. Konsep Dasar

Algoritma ini bekerja dengan tiga ide utama:

A. Haar Features

Fitur ini mirip filter hitam-putih (rectangular features). Membedakan area terang dan gelap pada citra (misalnya mata lebih gelap dari pipi).

Contoh fitur:

a. Fitur tepi (edge features)

b. Fitur garis (line features)

c. Fitur empat area (four-rectangle features)

Contohnya:

```
[ Putih | Hitam ] → mendeteksi perbedaan intensitas antara dua area
```

B. Integral Image

a. Trik cepat untuk menghitung jumlah piksel dalam area persegi panjang.

b. Ini bikin komputasi fitur jadi **super cepat**, bahkan untuk banyak jendela pencarian (sliding window).

C. AdaBoost Algorithm

a. Karena ada ribuan fitur, nggak semuanya penting.

b. AdaBoost memilih fitur terbaik dan membentuk **strong classifier** dari kombinasi **weak classifier**.

c. Hasilnya: sistem hanya pakai fitur paling informatif.

D. Cascade Classifier

a. Proses bertingkat: dari deteksi cepat → makin detail.

b. Setiap tahap (*stage*) menyaring citra yang **tidak** mengandung wajah.

c. Hanya area yang "lolos" dari semua stage yang dianggap sebagai wajah.

3. Tahapan Kerja Haar Cascade (untuk wajah)

a. **Input gambar/video**

b. **Konversi ke grayscale** (lebih ringan dan cukup untuk deteksi)

c. **Sliding window** mencari objek di berbagai ukuran

d. **Hitung Haar features** di tiap jendela

e. **Klasifikasi dengan Cascade**

f. **Tandai area yang lolos semua tahap** → wajah terdeteksi

4. Implementasi Praktis di Python (OpenCV)

```
import cv2

# Load classifier bawaan OpenCV
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')

# Baca gambar
img = cv2.imread('foto_wajah.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Deteksi wajah
faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5,
minSize=(30,30))

# Gambar kotak di sekitar wajah
for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)

cv2.imshow('Face Detection', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

5. Kelebihan & Kekurangan

Kelebihan	Kekurangan
Cepat & ringan	Kurang akurat untuk pencahayaan & sudut ekstrem
Cocok untuk real-time	Tidak tahan rotasi wajah
Bisa dilatih ulang	Butuh banyak data positif/negatif

6. Perbandingan Singkat

Algoritma	Jenis	Cocok untuk
Haar Cascade	Rule-based klasik	Real-time deteksi cepat
HOG + SVM	Fitur statistik	Pendeteksian wajah umum
DNN (CNN, MTCNN, SSD, YOLO)	Deep Learning	Deteksi multi-sudut, lebih akurat

7. Buat file haarcascade.html

Versi sederhana menampilkan deteksi wajah

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Real-time Face Detection with Haar Cascade (OpenCV.js)</title>
  <script async src="https://docs.opencv.org/4.x/opencv.js"
onload="onOpenCvReady();"></script>
  <style>
    body { text-align:center; font-family:sans-serif; background:#f4f4f4; }
    video, canvas {
      display:block;
      margin:10px auto;
      border-radius:8px;
      box-shadow:0 0 10px rgba(0,0,0,0.2);
    }
    #status { margin-top:10px; font-weight:bold; color:#333; }
  </style>
</head>
<body>
  <h2>Real-time Face Detection (Haar Cascade + OpenCV.js)</h2>
  <video id="video" width="640" height="480" autoplay muted></video>
  <canvas id="canvas" width="640" height="480"></canvas>
  <p id="status">Loading OpenCV...</p>

  <script>
    let video = document.getElementById("video");
    let canvas = document.getElementById("canvas");
    let ctx = canvas.getContext("2d");
    let streaming = false;
    let faceCascade, src, gray;

    function onOpenCvReady() {
      document.getElementById("status").innerText = "OpenCV.js ready!";
      startCamera();
    }

    function startCamera() {
      navigator.mediaDevices.getUserMedia({ video: true, audio: false })
      .then(stream => {
        video.srcObject = stream;
        video.play();
        video.addEventListener("playing", () => {
          streaming = true;
          initDetection();
        });
```

```
      })
      .catch(err => alert("Camera access denied: " + err));
    }

    function initDetection() {
      faceCascade = new cv.CascadeClassifier();
      const utils = new Utils('status');

      // load Haar Cascade XML
      const faceCascadeFile = 'haarcascade_frontalface_default.xml';
      utils.createFileFromUrl(faceCascadeFile, faceCascadeFile, () => {
        faceCascade.load(faceCascadeFile);
        startDetectionLoop();
      });
    }

    function startDetectionLoop() {
      src = new cv.Mat(video.height, video.width, cv.CV_8UC4);
      gray = new cv.Mat();
      const FPS = 30;

      function detect() {
        if (!streaming) {
          src.delete();
          gray.delete();
          return;
        }

        ctx.drawImage(video, 0, 0, video.width, video.height);
        src.data.set(ctx.getImageData(0, 0, video.width, video.height).data);
        cv.cvtColor(src, gray, cv.COLOR_RGBA2GRAY);

        let faces = new cv.RectVector();
        faceCascade.detectMultiScale(gray, faces, 1.1, 4, 0);

        for (let i = 0; i < faces.size(); ++i) {
          let face = faces.get(i);
          ctx.strokeStyle = "#00FF00";
          ctx.lineWidth = 3;
          ctx.strokeRect(face.x, face.y, face.width, face.height);
        }

        faces.delete();
        setTimeout(detect, 1000 / FPS);
      }
      detect();
    }

    // helper utils
    class Utils {
      constructor(print_id) { this.print = document.getElementById(print_id); }
      createFileFromUrl(path, url, callback) {
        let request = new XMLHttpRequest();
        request.open('GET', url, true);
        request.responseType = 'arraybuffer';
        request.onload = () => {
          if (request.status === 200) {
            let data = new Uint8Array(request.response);
            cv.FS_createDataFile('/', path, data, true, false, false);
            callback();
          } else {
            this.print.innerText = "✕ Gagal memuat " + url;
          }
        };
        request.send();
      }
    }
  </script>
</body>
</html>
```

8. **Versi menampilkan tiap proses Crop**, **Grayscale**, **Gaussian Blur**, **Edge (Canny)**, dan **LBP (Local Binary Pattern)**.

Tahap	Deskripsi
Crop	Memotong wajah dari frame berdasarkan hasil deteksi.
Grayscale	Mengubah dari RGB ke grayscale untuk efisiensi.
Blur	Mengurangi noise menggunakan Gaussian filter.
Edge	Menonjolkan tepi wajah dengan algoritma Canny.
LBP	Mengambil tekstur lokal wajah dengan Local Binary Pattern sederhana.

9. **Source code**

```
<!DOCTYPE html>
<html lang="id">
<head>
  <meta charset="utf-8">
  <title>Haar Cascade + Visualisasi Preprocessing</title>
  <style>
    body { font-family: Arial; background: #f4f4f4; text-align: center; }
    video, canvas { border: 1px solid #ccc; border-radius: 8px; margin: 5px; }
    .grid {
      display: flex;
      flex-wrap: wrap;
      justify-content: center;
      gap: 10px;
    }
  </style>
</head>
<body>
  <h2>Haar Cascade Face Detection + Preprocessing Visualization</h2>
  <video id="videoInput" width="320" height="240" autoplay muted></video>
  <canvas id="canvasOutput" width="320" height="240"></canvas>

  <h3>🧩 Tahapan Preprocessing</h3>
  <div class="grid">
    <div>
      <p>1️⃣ Crop</p>
      <canvas id="cropCanvas" width="100" height="100"></canvas>
    </div>
    <div>
      <p>2️⃣ Grayscale</p>
      <canvas id="grayCanvas" width="100" height="100"></canvas>
    </div>
    <div>
      <p>3️⃣ Blur</p>
      <canvas id="blurCanvas" width="100" height="100"></canvas>
    </div>
    <div>
      <p>4️⃣ Edge</p>
      <canvas id="edgeCanvas" width="100" height="100"></canvas>
    </div>
    <div>
      <p>5️⃣ LBP</p>
      <canvas id="lbpCanvas" width="100" height="100"></canvas>
    </div>
  </div>

  <!-- OpenCV.js -->
  <script async src="https://docs.opencv.org/4.x/opencv.js" onload="onOpenCvReady();"
type="text/javascript"></script>

  <script>
    async function onOpenCvReady() {
      console.log("✅ OpenCV.js siap digunakan");
      const video = document.getElementById("videoInput");
      const canvas = document.getElementById("canvasOutput");
```

```
      const ctx = canvas.getContext("2d");

      // Aktifkan kamera
      const stream = await navigator.mediaDevices.getUserMedia({ video: true });
      video.srcObject = stream;
      await video.play();

      // Load Haar Cascade
      const faceCascade = new cv.CascadeClassifier();
      const response = await
fetch("https://raw.githubusercontent.com/opencv/opencv/master/data/haarcascades/haarcasca
de_frontalface_default.xml");
      const data = await response.arrayBuffer();
      const dataView = new Uint8Array(data);
      const cascadeFile = "haarcascade_frontalface_default.xml";
      cv.FS_createDataFile("/", cascadeFile, dataView, true, false, false);
      faceCascade.load(cascadeFile);
      console.log("📂 Haarcascade loaded");

      // Variabel OpenCV
      let src = new cv.Mat(video.height, video.width, cv.CV_8UC4);
      let gray = new cv.Mat();
      let faces = new cv.RectVector();
      let cap = new cv.VideoCapture(video);

      function processVideo() {
        cap.read(src);
        cv.cvtColor(src, gray, cv.COLOR_RGBA2GRAY);

        // Deteksi wajah
        faceCascade.detectMultiScale(gray, faces, 1.1, 3, 0);

        ctx.drawImage(video, 0, 0, video.width, video.height);
        if (faces.size() > 0) {
          let face = faces.get(0);
          ctx.strokeStyle = "red";
          ctx.lineWidth = 2;
          ctx.strokeRect(face.x, face.y, face.width, face.height);

          // Crop wajah
          let faceROI = gray.roi(face);

          // 1️⃣ Crop
          cv.imshow("cropCanvas", faceROI);

          // 2️⃣ Grayscale (sudah gray)
          cv.imshow("grayCanvas", faceROI);

          // 3️⃣ Gaussian Blur
          let blur = new cv.Mat();
          cv.GaussianBlur(faceROI, blur, new cv.Size(5, 5), 0);
          cv.imshow("blurCanvas", blur);

          // 4️⃣ Edge Detection (Canny)
          let edges = new cv.Mat();
          cv.Canny(blur, edges, 80, 150);
          cv.imshow("edgeCanvas", edges);

          // 5️⃣ LBP (Local Binary Pattern sederhana)
          let lbp = new cv.Mat.zeros(faceROI.rows, faceROI.cols, cv.CV_8UC1);
          for (let i = 1; i < faceROI.rows - 1; i++) {
            for (let j = 1; j < faceROI.cols - 1; j++) {
              let center = faceROI.ucharPtr(i, j)[0];
              let code = 0;
              code |= (faceROI.ucharPtr(i - 1, j - 1)[0] > center) << 7;
              code |= (faceROI.ucharPtr(i - 1, j)[0] > center) << 6;
              code |= (faceROI.ucharPtr(i - 1, j + 1)[0] > center) << 5;
              code |= (faceROI.ucharPtr(i, j + 1)[0] > center) << 4;
              code |= (faceROI.ucharPtr(i + 1, j + 1)[0] > center) << 3;
              code |= (faceROI.ucharPtr(i + 1, j)[0] > center) << 2;
              code |= (faceROI.ucharPtr(i + 1, j - 1)[0] > center) << 1;
              code |= (faceROI.ucharPtr(i, j - 1)[0] > center) << 0;
              lbp.ucharPtr(i, j)[0] = code;
            }
```

```
          }
          cv.imshow("lbpCanvas", lbp);

          // Bersihkan
          faceROI.delete(); blur.delete(); edges.delete(); lbp.delete();
        }
        requestAnimationFrame(processVideo);
      }
      requestAnimationFrame(processVideo);
    }
  </script>
</body>
</html>
```

Pertemuan selanjutnya **proyek kecil (mini project Haar Cascade)** yang cocok untuk mahasiswa atau riset tingkat awal — misalnya:

1. Face + Eye detection
2. Smile detection system
3. Vehicle detection
4. Face & mask detection kombinasi