Practical Assignment No.4

Face Detection using Viola-Jones Approach

Member 1
Name: Zhou Xuze
ITMO number: 321528
HDU number: 20322332

Member 2
Name: Wang Tianyu
ITMO number: 321520
HDU number: 20322324

I . Purpose

Study of Viola-Jones approach for detection of faces and part of bodies in the images.
be familiar with the functions of the MATLAB for working with the cascade object detectors and Viola-Jones approach.

II . Theoretical substantiation

The Viola-Jones face detector method is based on the following concepts:
1. Haar-like features as weak classifiers.
2. Integral image representation for fast calculation of haar-like features.
3. AdaBoost training method to combine weak classifiers into a strong classifier.
4. Combining of strong classifiers into a cascade classifier.

Haar-like feature is a kind of a weak classifier. It can be defined as the difference of the sum of pixels of areas inside the rectangle, which can be at any position and scale within the original image. In a traditional Viola-Jones face detector algorithm 4 types of Haar-like features are used. To calculate the value of the Haar-like feature we need to calculate sums of pixels inside rectangular areas of the image and do it as fast as possible.

K iterations: $\epsilon_k = P_{r_i D_k}$

$$\alpha_k = 0.5 * \ln\left(\frac{1-\epsilon_k}{\epsilon_k}\right)$$

$$D_{k+1}(i) = \frac{D_k(i)}{Z_k} * \begin{cases} e^{-\alpha_k}, h_k(x_i) = y_i \\ e^{\alpha_k}, h_k(x_i) \neq y_i \end{cases}$$

$$H(x) = sign\left(\sum_{k=1}^{K} \alpha_k * h_k(x)\right)$$

Cascade classifiers: A strong classifier that have a required accuracy may require calculation of too much weak classifiers that would slow down the detection speed taking into an account that most of scanned windows do not contain faces. To speed up the detection rate a set of classifiers with increasing complexity are organized in a cascade of classifiers. The cascade contains a set of classifiers with an increasing complexity and detection rate.

To be classified positively, a sliding window should pass all cascade stages. In case if any classifier rejects the window, it is immediately rejected and detector proceeds to the next window. As a result, that most of negative windows are rejected fast with first fast classifiers in the cascade.

$$TP = \prod_i T\,P_i$$

$$FP = \prod_i F\,P_i$$

III. Assignment progress

1. Faces detection

(a) Original images

Picture 1

Picture 2

Picture 3

(b) Code of the scripts

```
faceDetector = vision.CascadeObjectDetector;
I = imread('face1.png');
bboxes = faceDetector(I);
IFaces = insertObjectAnnotation(I, 'rectangle', bboxes, 'Face');
imshow(IFaces)
```

(c) Resulting images.

Picture 1

Picture 2

Picture 3

2. Body parts detection

(a) Original image

Picture 1

Picture 2

Picture 3

(b) Code of the scripts

```
% eyes
eyesDetector = vision.CascadeObjectDetector('ClassificationModel',
'EyePairSmall', 'UseROI', true);
counter = 1;
for i = 1:1:length(bboxes)
    temp = eyesDetector(I, bboxes(i, :));
    if ~isempty(temp)
        bboxes2(counter, :) = temp;
        counter = counter + 1;
    end
end
IEyes = insertObjectAnnotation(IFaces, 'rectangle', bboxes2, 'Eyes',
'Color', 'magenta');
imshow(IEyes);


% nose
noseDetector = vision.CascadeObjectDetector('ClassificationModel', 'Nose',
'UseROI', true);
counter = 1;
for i = 1:1:length(bboxes)
    temp = noseDetector(I, bboxes(i, :));
    if ~isempty(temp)
        bboxes3(counter, :) = temp;
        counter = counter + 1;
    end
end
INose = insertObjectAnnotation(IFaces, 'rectangle', bboxes3, 'Nose',
'Color', 'blue');
imshow(INose);
```

(c) Resulting images

Picture 1

① eyes

② noise

Picture 2

① eyes

② noise

Picture 3

① eyes

② noise

3. Optional 1

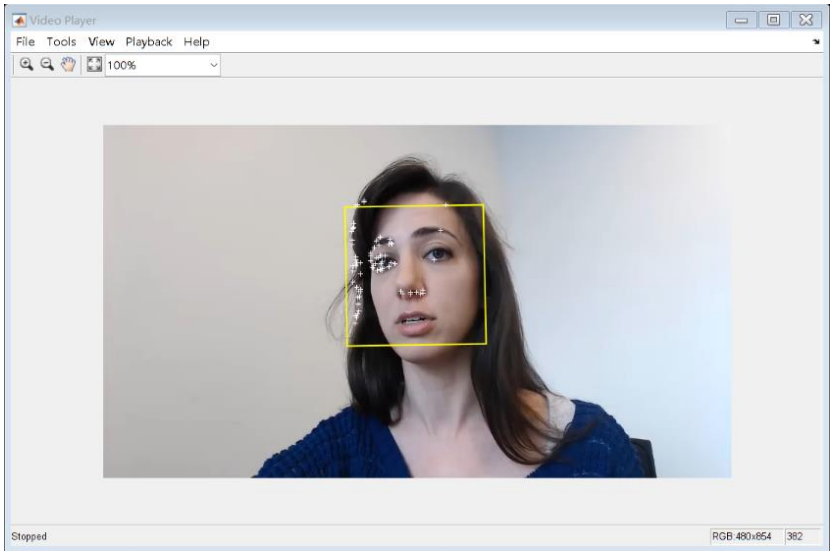
(a) Original image

(b) Code of the scripts

```
faceDetector = vision.CascadeObjectDetector();
videoReader = VideoReader('face.mp4');
videoframe = readFrame(videoReader);
bbox = step(faceDetector, videoFrame);
videoFrame = insertShape(videoFrame, 'Rectangle', bbox);
figure; imshow(videoFrame); title('Detected face');
bboxPoints = bbox2points(bbox(1, :));
points = detectMinEigenFeatures(rgb2gray(videoFrame), 'ROI', bbox);
figure, imshow(videoFrame), hold on, title('Detected features');
plot(points);
points = points.Location;
initialize(pointTracker, points, videoFrame);
videoPlayer  = vision.VideoPlayer('Position',[100 100 [size(videoFrame, 2),
size(videoFrame, 1)]+30]);
oldPoints = points;
while hasFrame(videoReader)
    % Get the next frame
    videoFrame = readFrame(videoReader);
    % Track the points. Note that some points may be lost.
    [points, isFound] = step(pointTracker, videoFrame);
    visiblePoints = points(isFound, :);
```

```
    oldInliers = oldPoints(isFound, :);

    if size(visiblePoints, 1) >= 2
        % Estimate the geometric transformation between the old points
        % and the new points and eliminate outliers
        [xform, oldInliers, visiblePoints] =
estimateGeometricTransform(oldInliers, visiblePoints, 'similarity',
'MaxDistance', 4);

        % Apply the transformation to the bounding box points
        bboxPoints = transformPointsForward(xform, bboxPoints);

        % Insert a bounding box around the object being tracked
        bboxPolygon = reshape(bboxPoints', 1, []);
        videoFrame = insertShape(videoFrame, 'Polygon', bboxPolygon, ...
            'LineWidth', 2);

        % Display tracked points
        videoFrame = insertMarker(videoFrame, visiblePoints, '+','Color',
'white');
        % Reset the points
        oldPoints = visiblePoints;
        setPoints(pointTracker, oldPoints);
    end
        % Display the annotated video frame using the video player object
    step(videoPlayer, videoFrame);
end
release(videoPlayer);
```

(c) Resulting image



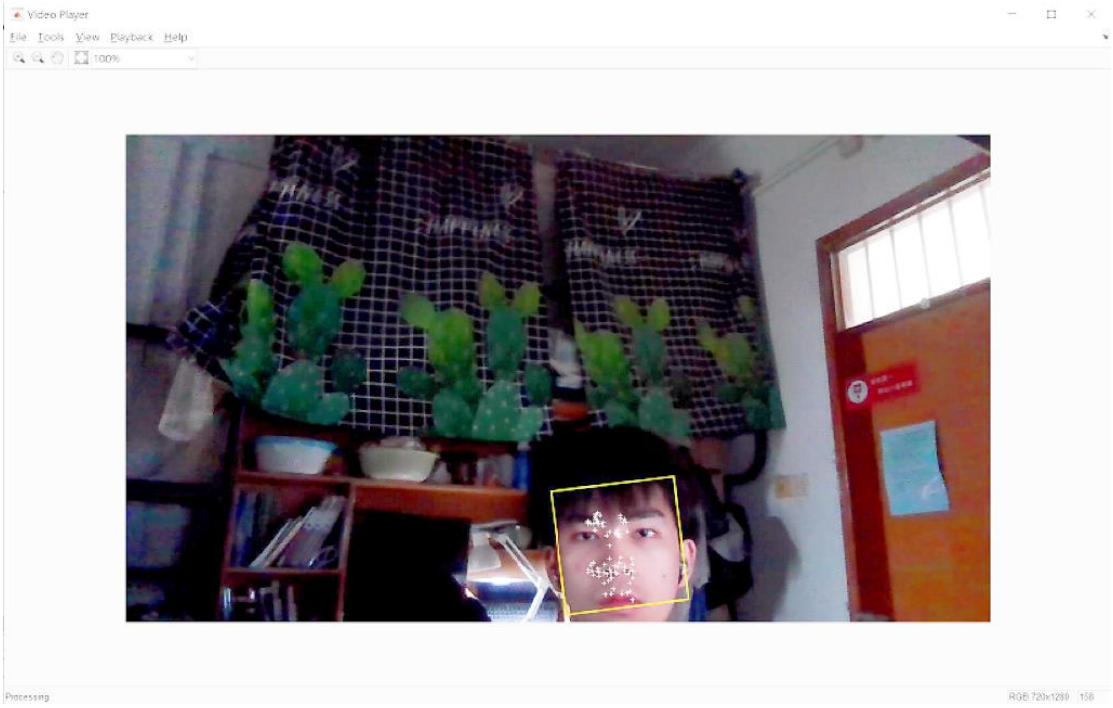
4. Optional 2

(a) Code of the scripts

```
faceDetector = vision.CascadeObjectDetector();

% Create the point tracker object.
pointTracker = vision.PointTracker('MaxBidirectionalError', 2);

% Create the webcam object.
cam = webcam();

% Capture one frame to get its size.
videoFrame = snapshot(cam);
frameSize = size(videoFrame);

% Create the video player object.
videoPlayer = vision.VideoPlayer('Position', [100 100 [frameSize(2),
frameSize(1)]+30]);

runLoop = true;
numPts = 0;
frameCount = 0;

while runLoop && frameCount < 400

    % Get the next frame.
    videoFrame = snapshot(cam);
    videoFrameGray = rgb2gray(videoFrame);
    frameCount = frameCount + 1;

    if numPts < 10
        bbox = faceDetector.step(videoFrameGray);

        if ~isempty(bbox)
            points = detectMinEigenFeatures(videoFrameGray, 'ROI',
bbox(1, :));
            xyPoints = points.Location;
            numPts = size(xyPoints,1);
            release(pointTracker);
            initialize(pointTracker, xyPoints, videoFrameGray);
            % Save a copy of the points.
            oldPoints = xyPoints;
            bboxPoints = bbox2points(bbox(1, :));
```

```
            bboxPolygon = reshape(bboxPoints', 1, []);

            % Display a bounding box around the detected face.
            videoFrame = insertShape(videoFrame, 'Polygon', bboxPolygon,
'LineWidth', 3);

            % Display detected corners.
            videoFrame = insertMarker(videoFrame, xyPoints, '+', 'Color',
'white');
        end
    else
        % Tracking mode.
        [xyPoints, isFound] = step(pointTracker, videoFrameGray);
        visiblePoints = xyPoints(isFound, :);
        oldInliers = oldPoints(isFound, :);
        numPts = size(visiblePoints, 1);

        if numPts >= 10
            % Estimate the geometric transformation between the old points and
the new points.
            [xform, oldInliers, visiblePoints] =
estimateGeometricTransform(oldInliers, visiblePoints, 'similarity',
'MaxDistance', 4);
            bboxPoints = transformPointsForward(xform, bboxPoints);
            bboxPolygon = reshape(bboxPoints', 1, []);
            % Display a bounding box around the face being tracked.
            videoFrame = insertShape(videoFrame, 'Polygon', bboxPolygon,
'LineWidth', 3);
            % Display tracked points.
            videoFrame = insertMarker(videoFrame, visiblePoints, '+', 'Color',
'white');
            % Reset the points.
            oldPoints = visiblePoints;
            setPoints(pointTracker, oldPoints);
        end
    end
    % Display the annotated video frame using the video player object.
    step(videoPlayer, videoFrame);
    runLoop = isOpen(videoPlayer);
end
clear cam;
release(videoPlayer);
release(pointTracker);
release(faceDetector);
```

(b) Resulting image



IV. Question (Conclusion)

1. What is the special image representation used in the Viola-Jones approach?

The special image representation is integral Image. The feature calculation is sped up by using an integral image representation. Each pixel in this representation holds the total of all the pixels' values that are above and to the left of it.

2. What is the main advantage of Haar-like features for classifier training?

The benefit of using Haar like is that it can more accurately characterize changes in light and shadow. It is utilized to identify the positive face as a result. This explains why Haar is clearly more effective when used for positive face detection than for side face detection: the presence of protrusions like the nose makes the light and shadow shifts on the face more evident. In addition, it can help to speed up the calculation process.

3. Could you use Viola-Jones approach for detecting arbitrary objects and why?

No, because there are still issues with the Viola-Jones algorithm. A relatively straightforward and unstable feature is the harr-like functionality. Simple decision trees, which are easier to overfit, are used by the weak classifier. As a result, the algorithm effectively solves the front face. The processing outcome is not optimal for the unique and complex scenarios like facial occlusion, posture, and expression. The outcome can be worse if it's used to find additional things.

Additional conclusions

① Viola-Jones approach overcomes the inefficiency caused by sliding window detection and can be used for real-time face detection. More importantly, it presents a general idea to solve the problem of target detection.

② There still exists some problems with Viola-Jones approach. For instance, some features will be missed when we use this algorithm to detect face or some other body parts (The algorithm has the problem of missing the detection of faces with deviated angles and false detection of non-face areas in the surrounding environment).