ДОДАТОК А

Програмний код

```
# parameters for loading data and images
emotion_model_path = './models/emotion_model.hdf5'
emotion_labels = get_labels('fer2013')

# hyper-parameters for bounding boxes shape
frame_window = 10
emotion_offsets = (20, 40)

# loading models
face_cascade = cv2.CascadeClassifier('./models/haarcascade_frontalface_default.xml')
emotion_classifier = load_model(emotion_model_path)

# getting input model shapes for inference
emotion_target_size = emotion_classifier.input_shape[1:3]

# starting lists for calculating modes
emotion_window = []

# starting video streaming

cv2.namedWindow('window_frame')
video_capture = cv2.VideoCapture(0)

# Select video or webcam feed
cap = None
   cap = cv2.VideoCapture(0) # Webcam source
else:
   cap = cv2.VideoCapture('./demo/dinner.mp4') # Video file source

while cap.isOpened(): # True:
   ret, bgr_image = cap.read()

   #bgr_image = video_capture.read()[1]

   gray_image = cv2.cvtColor(bgr_image, cv2.COLOR_BGR2GRAY)
   rgb_image = cv2.cvtColor(bgr_image, cv2.COLOR_BGR2RGB)

   faces = face_cascade.detectMultiScale(gray_image, scaleFactor=1.1, minNeighbors=5,
       minSize=(30, 30), flags=cv2.CASCADE_SCALE_IMAGE)

   for face_coordinates in faces:

      x1, x2, y1, y2 = apply_offsets(face_coordinates, emotion_offsets)
      gray_face = gray_image[y1:y2, x1:x2]
      try:
         gray_face = cv2.resize(gray_face, (emotion_target_size))
      except:
```

```
            continue

        gray_face = preprocess_input(gray_face, True)
        gray_face = np.expand_dims(gray_face, 0)
        gray_face = np.expand_dims(gray_face, -1)
        emotion_prediction = emotion_classifier.predict(gray_face)
        emotion_probability = np.max(emotion_prediction)
        emotion_label_arg = np.argmax(emotion_prediction)
        emotion_text = emotion_labels[emotion_label_arg]
        emotion_window.append(emotion_text)

        if len(emotion_window) > frame_window:
            emotion_window.pop(0)
        try:
            emotion_mode = mode(emotion_window)
        except:
            continue

        if emotion_text == 'angry':
            color = emotion_probability * np.asarray((255, 0, 0))
        elif emotion_text == 'sad':
            color = emotion_probability * np.asarray((0, 0, 255))
        elif emotion_text == 'happy':
            color = emotion_probability * np.asarray((255, 255, 0))
        elif emotion_text == 'surprise':
            color = emotion_probability * np.asarray((0, 255, 255))
        else:
            color = emotion_probability * np.asarray((0, 255, 0))

        color = color.astype(int)
        color = color.tolist()

        draw_bounding_box(face_coordinates, rgb_image, color)
        draw_text(face_coordinates, rgb_image, emotion_mode,
                  color, 0, -45, 1, 1)

    bgr_image = cv2.cvtColor(rgb_image, cv2.COLOR_RGB2BGR)
    cv2.imshow('window_frame', bgr_image)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
def extract_pulse(rPPG,fftlength,fs_video):

    if(rPPG.shape[1] < fftlength):
        return np.zeros(int(fftlength/2)+1)
    fft_roi = range(int(fftlength/2+1))
    bpf_div= 60 * fs_video / 2
    b_BPF40220,a_BPF40220 = signal.butter(10, ([40/bpf_div, 220/bpf_div]),  'bandpass')
```

```
    col_c = np.zeros((3,fftlength))
    skin_vec = [1,0.66667,0.5]
    for col in [R,G,B]:
        col_stride = rPPG[col,-fftlength:]# select last samples
        y_ACDC = signal.detrend(col_stride/np.mean(col_stride))
        col_c[col] = y_ACDC * skin_vec[col]
    X_chrom = col_c[R]-col_c[G]
    Y_chrom = col_c[R] + col_c[G] - 2* col_c[B]
    Xf = signal.filtfilt(b_BPF40220,a_BPF40220,X_chrom)
    Yf = signal.filtfilt(b_BPF40220,a_BPF40220,Y_chrom)
    Nx = np.std(Xf)
    Ny = np.std(Yf)
    alpha_CHROM = Nx/Ny
    x_stride_method = Xf- alpha_CHROM*Yf
    STFT = np.fft.fft(x_stride_method,fftlength)[fft_roi]
    normalized_amplitude = np.abs(STFT)/np.max(np.abs(STFT))
    return normalized_amplitude

from scipy.spatial import distance as dist
from imutils.video import VideoStream
from imutils import face_utils
import argparse
import imutils
import time
import dlib
import cv2

def eye_aspect_ratio(eye):


   A = dist.euclidean(eye[1], eye[5])
   B = dist.euclidean(eye[2], eye[4])



   C = dist.euclidean(eye[0], eye[3])


   ear = (A + B) / (2.0 * C)


   return ear
EYE_AR_THRESH = 0.23


EYE_AR_CONSEC_FRAMES = 1


COUNTER = 0
TOTAL = 0
```

```
print("[INFO] loading facial landmark predictor...")
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")


(lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]


print("[INFO] starting video stream thread...")

fileStream = True
vs = VideoStream(src=0).start()

fileStream = False
time.sleep(1.0)


BLINKS = 0

start_time = time.time()

while True:


    if fileStream and not vs.more():
        break


    frame = vs.read()
    frame = imutils.resize(frame, width=450)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)


    rects = detector(gray, 0)


    for rect in rects:


        shape = predictor(gray, rect)
        shape = face_utils.shape_to_np(shape)


        leftEye = shape[lStart:lEnd]
        rightEye = shape[rStart:rEnd]
        leftEAR = eye_aspect_ratio(leftEye)
        rightEAR = eye_aspect_ratio(rightEye)


        ear = (leftEAR + rightEAR) / 2.0
```

```
        leftEyeHull = cv2.convexHull(leftEye)
        rightEyeHull = cv2.convexHull(rightEye)
        cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0), 1)
        cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1)


        if ear < EYE_AR_THRESH:
            COUNTER += 1
        else:

            if COUNTER >= EYE_AR_CONSEC_FRAMES:
                TOTAL += 1
                BLINKS+=1
            COUNTER = 0


        cv2.putText(frame, "Blinks: {}".format(TOTAL), (10, 30),cv2.FONT_HERSHEY_SIMPLEX,
0.7, (0, 0, 255), 2)
        cv2.putText(frame, "EAR: {:.2f}".format(ear), (300, 30),cv2.FONT_HERSHEY_SIMPLEX, 0.7,
(0, 0, 255), 2)

        end_time = time.time()
        if(end_time - start_time > 60):
            if(BLINKS < 10):
                print("Blink now !")
            else:
                print("All okay")

            start_time = time.time()
            BLINKS=0


    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF


    if key == ord("q"):
        break


cv2.destroyAllWindows()
vs.stop()
import numpy as np
import cv2

import matplotlib.pyplot as plt

from scipy import interpolate, signal, optimize
from scipy.fftpack import fft, ifft, fftfreq, fftshift

from sklearn.decomposition import PCA
```

```
class SignalProcess:

    def __init__(self, signal_source, fs=30, draw=False):

        self.signal_source = signal_source
        self.fs = fs

        self.bpm_list = []
        self.mean_bpm = 0

        self.draw = draw

        # Graphs
        self.fig, (self.ax1, self.ax2) = plt.subplots(2, 1)
        self.graph_message = ''

        if self.draw:
            self.fig.show()

    def get_diffs(self, traces, fps):
        # Filter traces get 4sec or long nyquist freq for 0.5 hz -> 1 hZ
        traces = [trace for trace in traces if len(trace) > 2*fps]
        trace_max_len = max( [len(trace) for trace in traces] )

        #trace_max_len = 300
        #TODO: This is quickfix
        traces = [trace for trace in traces if len(trace) == trace_max_len]

        # Calculate y movement of each
        displacements = []
        #displacements = np.array([[]])
        for trace in traces:
            trace = np.array(trace)

            y_pts = trace[:, 1]
            # Pad array to standart lenght
            len_diff = trace_max_len-len(y_pts)
            if len_diff > 0:
                pass
                print('Padded', len_diff)
            y_pts = np.pad(y_pts, (len_diff, 0), 'edge')

            displace = np.diff(y_pts) # y coordinates
            displacements.append(displace)

        if len(displacements) > 0:
            displacements = np.stack(displacements, axis=0)

        return displacements
```

```
def get_y(self, traces):
    """ Get Y coordinates of given traces """
    # Filter traces get 4sec or long nyquist freq for 0.5 hz -> 1 hZ
    traces = [trace for trace in traces if len(trace) > 2*self.fs]
    trace_max_len = max( [len(trace) for trace in traces] )

    #trace_max_len = 300
    #TODO: This is quickfix
    traces = [trace for trace in traces if len(trace) == trace_max_len]

    # Calculate y movement of each
    ys = []
    #displacements = np.array([[]])
    for trace in traces:
        trace = np.array(trace)[:, 1]

        ys.append(trace)
    return np.stack(ys, axis=0)

def filter_signal(self, signal_data, fs=30, low_c=0.75, high_c=2.0):
    """
    This function bandpass filters given signal
    :param signal_data: Input Signal
    :param fs: Sampling Frequency
    :param low_c: Low Cutoff Frequency
    :param high_c: High Cutoff Frequency
    :return: Filtered signal
    """

    # number of signal points
    N = len(signal_data)
    # sample spacing
    T = 1.0 / fs

    #Draw signal
    #t = np.arange(len(displace))/fps
    t = np.linspace(0.0, T*N, N)

    # Filter signal
    fc = np.array([low_c, high_c])  # Cut-off frequency of the filter
    # 0.75 hz - 2 hz => 45bpm - 120bpm

    w = fc / (fs / 2) # Normalize the frequency
    b, a = signal.butter(5, w, 'bandpass')

    filter_output = signal.filtfilt(b, a, signal_data)

    return filter_output

def filter_out(self, signals, low_c=0.5, high_c=2.0):
    filtered_signals = []
```

```
    for signal_data in signals:
        filter_out = self.filter_signal(signal_data, fs=self.fs, low_c=low_c, high_c=high_c)
        filtered_signals.append(filter_out)

    if len(filtered_signals) > 0:
        filtered_signals = np.stack(filtered_signals, axis=0)

    return filtered_signals[:-self.fs]

def get_mean(self, filtered_signals, show=True):
    if len(filtered_signals) < 5:
        return 0

    mean_signal = np.mean(filtered_signals, axis=0, dtype=np.float64)
    maxFreq, percentage = self.get_dominant_frequency(mean_signal, fs=self.fs, draw=False)

    bpm = maxFreq * 60

    if show:
        self.ax1.cla()
        self.ax2.cla()
        self.get_dominant_frequency(mean_signal, fs=self.fs, draw=True)
        self.fig.canvas.draw()

    return bpm


def get_dominant_frequency(self, signal_data, fs=30, draw=False):
    # number of signal points
    N = len(signal_data)
    # sample spacing
    T = 1.0 / fs

    # Get fft
    spectrum = np.abs(fft(signal_data))
    spectrum *= spectrum
    xf = fftfreq(N, T)

    # Get maximum ffts index from second half
    #maxInd = np.argmax(spectrum[:int(len(spectrum)/2)+1])
    maxInd = np.argmax(spectrum)
    maxFreqPow = spectrum[maxInd]
    maxFreq = np.abs(xf[maxInd])

    total_power = np.sum(spectrum)
    # Get max frequencies power percentage in total power
    percentage = maxFreqPow / total_power

    if draw:
        t = np.linspace(0.0, T*N, N)

        self.ax1.set_title('Signal data')
```

```
        self.ax1.plot(t, signal_data)
        #self.ax1.plot(peaks/fps, signal_data[peaks], "x")
        #self.ax1.plot(np.zeros_like(t/fps), "--", color="gray")
        self.ax1.set(xlabel='Time', ylabel='Pixel movement')
        self.ax1.grid()

        self.ax2.plot(xf, 1.0/N * spectrum)
        self.ax2.set_title('FFT')
        self.ax2.axvline(maxFreq, color='red')
        self.ax2.grid()
        self.ax2.set(xlabel='Freq', ylabel='')

        #print("Max power Freq {} % {} BPM:{}".format(maxFreq, percentage, bpm))

    return maxFreq, percentage


def do_pca(self, filtered_signals, fps, show=True):
    """
     Reduces signals into 5 channels by PCA then finds dominant frequency on each of them and
     selects the most dominant one based on how much power it has on total power
     """
    if len(filtered_signals) < 5:
        return 0

    pca = PCA(n_components=5)
    pca_result = pca.fit_transform(filtered_signals.T).T

    max_ratios = []
    max_freqs = []
    for i, signal_data in enumerate(pca_result):
        maxFreq, percentage = self.get_dominant_frequency(signal_data, fs=fps, draw=False)
        max_ratios.append(percentage)
        max_freqs.append(maxFreq)

    # Find most dominant out of pcas
    idx = np.argmax(max_ratios)
    last_pca = pca_result[idx]

    self.graph_message = "Selected PCA:{}".format(idx)

    bpm = max_freqs[idx]*60

    if show:
        self.ax1.cla()
        self.ax2.cla()
        self.get_dominant_frequency(last_pca, fs=fps, draw=True)
        self.fig.canvas.draw()
    return bpm

def find_bpm(self, bpm_list_len=10, low_c=0.5, high_c=3.0):
```

```
        bpm = 0

        traces = self.get_y(self.signal_source.traces)
        filtered_signals = self.filter_out(traces, low_c=low_c, high_c=high_c)

        # If finding bpm from face shape
        if self.signal_source.face.dedector_type == 'face_shape':
            bpm = self.get_mean(filtered_signals, self.draw)
        else:
            bpm = self.do_pca(filtered_signals, self.fs, show=self.draw)

        self.bpm_list.insert(0, bpm)

        if len(self.bpm_list) > bpm_list_len:
            self.bpm_list.pop()

        self.mean_bpm = sum(self.bpm_list) / len(self.bpm_list)

        return bpm
import numpy as np
import cv2


def draw_str(dst, target, s):
    x, y = target
    cv2.putText(dst, s, (x+1, y+1), cv2.FONT_HERSHEY_PLAIN, 1.0, (0, 0, 0), thickness = 2,
lineType=cv2.LINE_AA)
    cv2.putText(dst, s, (x, y), cv2.FONT_HERSHEY_PLAIN, 1.0, (255, 255, 255),
lineType=cv2.LINE_AA)


class TrackPoints:
    def __init__(self, face_dedector, max_trace_num=150, max_trace_history=60):

        self.traces = []
        self.max_trace_num = max_trace_num
        self.max_trace_history = max_trace_history
        self.track_started = False

        self.lastest_points = []

        self.face = face_dedector

        self.lk_params = dict( winSize  = (15,15),
                maxLevel = 2,
                criteria = (cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 0.03))


    def get_first_points(self, prev_frame, curr_frame):
        track_point_candidates = self.face.get_points_pipeline(prev_frame)

        if track_point_candidates is not None:
```

```
            if self.face.dedector_type == 'face_shape':
                initial_points = track_point_candidates
            else:
                initial_points = self.filter_unbacktrackable(prev_frame, curr_frame, track_point_candidates)

            # Add initial points
            for i, (x, y) in enumerate(np.float32(initial_points).reshape(-1, 2)):
                if i < self.max_trace_num:
                    self.traces.append([(x, y)])

            self.track_started = True

    def filter_unbacktrackable(self, prev_frame, curr_frame, track_point_candidates,
ret_nextPts=False):
        if len(track_point_candidates) < 1:
            if not ret_nextPts:
                return []
            else:
                return [], []

        #Forward optical flow
        nextPts, st, err = cv2.calcOpticalFlowPyrLK(prev_frame, curr_frame, track_point_candidates,
None, **self.lk_params)
        # Backward optical flow
        backNextPts, _st, _err = cv2.calcOpticalFlowPyrLK(curr_frame, prev_frame, nextPts, None,
**self.lk_params)

        # Find differance between 2 estimates
        dist = abs(track_point_candidates-backNextPts).reshape(-1, 2).max(-1)

        # Select backtraced points that are in 1 pixel dist
        bool_filter = dist < 1

        if not ret_nextPts:
            return track_point_candidates[bool_filter.flatten()]
        else:
            return nextPts, bool_filter

    def add_new_traces(self, prev_frame, curr_frame):
        track_point_candidates = self.face.get_points_pipeline(prev_frame)
        if track_point_candidates is not None:
            initial_points = self.filter_unbacktrackable(prev_frame, curr_frame, track_point_candidates)

            # Add initial points
            for x, y in np.float32(initial_points).reshape(-1, 2):
                if len(self.traces) < self.max_trace_num:
                    # Check if same as another point
                    if not [x,y] in self.lastest_points:
                        self.traces.append([(x, y)])

    def filter_none_face(self, curr_frame):
```

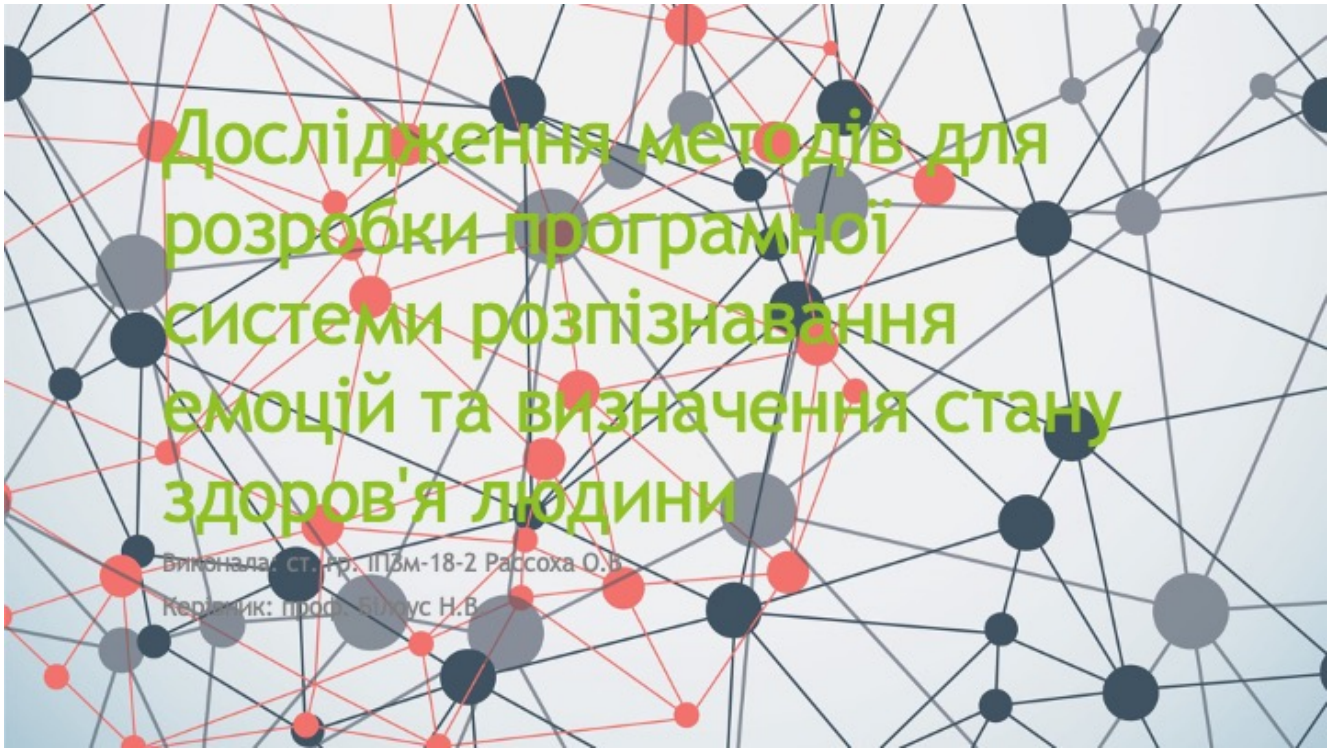
```
        # Update face rectangle
        face_rect = self.face.detect_face(curr_frame)
        mask = self.face.get_roi_mask(curr_frame, face_rect)

        if face_rect is not None:
            new_lastest_points = []
            new_traces = []
            for i, (x,y) in enumerate(self.lastest_points):

                # If inside of face region add new list
                if self.face.point_in_rectangle(x,y, *self.face.face_rectange):
                    if not self.face.point_in_rectangle(x,y, *self.face.eyes_rectangle):
                        #print(i, 'in face')
                        new_lastest_points.append(self.lastest_points[i])
                        new_traces.append(self.traces[i])

            self.lastest_points = new_lastest_points
            self.traces = new_traces


    def track_points(self, prev_frame, curr_frame):
        if not self.track_started:
            self.get_first_points(prev_frame, curr_frame)
            if not self.track_started:
                return

        # Get previous frames from traces
        prevPts = np.float32([tr[-1] for tr in self.traces]).reshape(-1, 1, 2)
        nextPts, bool_filter = self.filter_unbacktrackable(prev_frame, curr_frame, prevPts,
ret_nextPts=True)
        nextPts = nextPts.reshape(-1, 2)

        # Reset tracking
        if len(nextPts) < 1:
            self.track_started = False
            return

        # TODO: a hacky implementation
        if self.face.dedector_type == 'face_shape':

            # Get face points
            points = self.face.get_points_pipeline(curr_frame)
            points = np.array(points)

            # check if every point is at (0,0)
            if (points == 0).all():
                # Face is not dedected use lastest points
                return

            # Check untracable points and replace them with facepoints (int)
            idx = np.where(bool_filter == False)
            nextPts[idx] = points[idx]
```

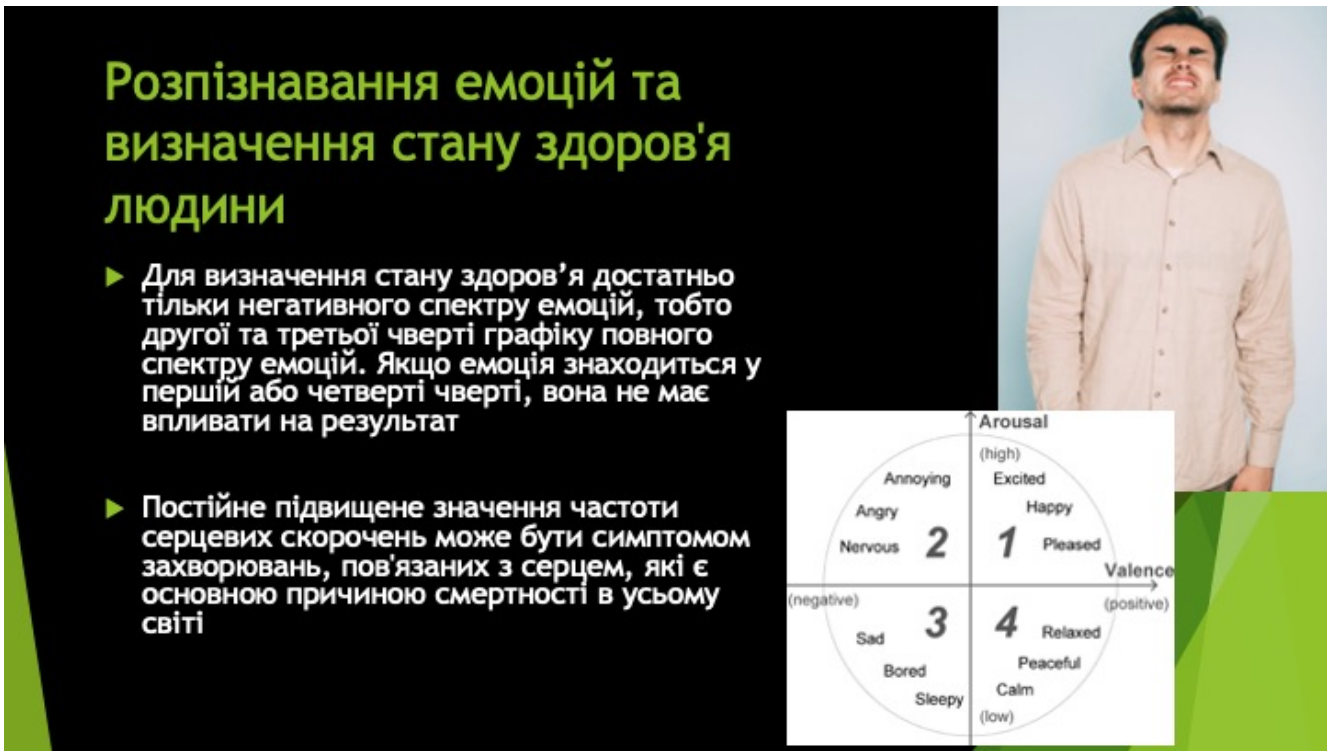
```
            new_traces = []
            self.lastest_points = []
            # add from starting point
            for trace, (x,y) in zip(self.traces, nextPts):

                trace.append((x,y))
                self.lastest_points.append([x, y])

                # If trace history gets too big delete oldest element
                if len(trace) > self.max_trace_history:
                    del trace[0]

                new_traces.append(trace)

            self.traces = new_traces
        else:

            new_traces = []
            self.lastest_points = []
            # add from starting point
            for trace, (x,y), good_flag in zip(self.traces, nextPts.reshape(-1, 2), bool_filter):
                # Delete unbacktrackable traces
                if not good_flag:
                    continue

                trace.append((x,y))
                self.lastest_points.append([x, y])

                # If trace history gets too big delete oldest element
                if len(trace) > self.max_trace_history:
                    del trace[0]

                new_traces.append(trace)

            self.traces = new_traces

            # Filter out points outside face mask
            #self.filter_none_face(curr_frame)

            # Add new traces if it shrink
            if len(self.traces) < self.max_trace_num:
                self.add_new_traces(prev_frame, curr_frame)


    def get_current_points(self):
        return np.int32([tr[-1] for tr in self.traces]).reshape(-1, 1, 2)


if __name__ == "__main__":
```

```
capture = cv2.VideoCapture(0)
#capture = cv2.VideoCapture('./data/face_videos/sitting2.avi')
frame_c = 0
gray_frames = [] #0 is newest -1 is oldest

face = FacePoints(dedector_type='face_shape')
tracking = TrackPoints(face_dedector=face)

# Create some random colors
color = np.random.randint(0,255,(100,3))


while capture.isOpened():
    # getting a frame
    ret, frame = capture.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    vis = frame.copy()

    gray_frames.insert(0, gray)


    # Wait 10 frames before selecting points
    if frame_c >= 10:
        gray_frames.pop()

        tracking.track_points(gray_frames[1], gray_frames[0])
        nextPts = tracking.get_current_points()

        # Draw points
        for i, new in enumerate(nextPts):
            a,b = new.ravel()
            vis = cv2.circle(vis,(a,b),5,color[i%100].tolist(),-1)

        # Draw Tracks
        cv2.polylines(vis, [np.int32(tr) for tr in tracking.traces], False, (0, 255, 0))
        draw_str(vis, (20, 20), 'trace count: %d' % len(tracking.traces))
    # Show
    cv2.imshow('Track points', vis)
    if cv2.waitKey(1) == 27:
        break
    frame_c += 1

capture.release()
cv2.destroyAllWindows()
```

ДОДАТОК Б

Слайди презентації





Постановка задачі

Поставлена задача провести дослідження методів визначення ЧС та розпізнавання емоцій. Розробити систему відслідковування фізичного та емоційного стану людини на прикладі додатку для спортзалу.

Вимоги до системи

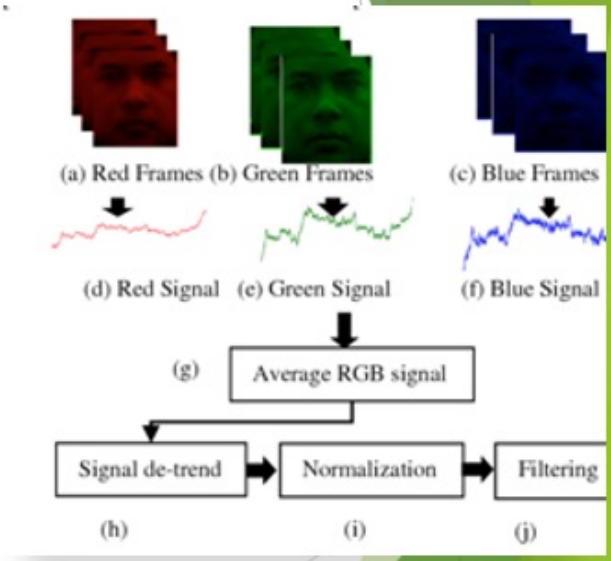
- система повинна надавати інформацію про клієнтів, відображати загальний стан здоров'я кожного з клієнтів, наявність або відсутність болю;
- система повинна надавати інформацію про дані збережені з камер, відображати загальний стан здоров'я клієнтів у момент часу у разі виявлення відхилень від норми;
- система повинна надавати інформацію про частоту серцебиття користувача;

Актуальність

Практичні результати магістерської роботи можуть бути для розпізнавання емоцій та визначення стану здоров'я людини на прикладі додатку для спортивного залу, який стежить за станом клієнтів у момент часу, зберігає дані про клієнтів та дає змогу адміністраторам переглянути статистику та зробити висновки щодо того, хто з клієнтів перенапружується або відчуває біль під час тренувань. А, отже попередити перенавантаження та травми або вчасно відреагувати на небезпечну ситуацію.

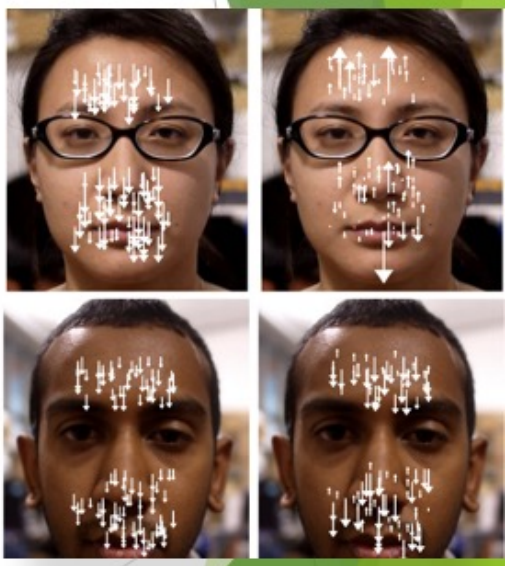
Аналіз методів визначення частоти сердцевих здоров'я людини

- Методи дистанційної фотоплетізмографії



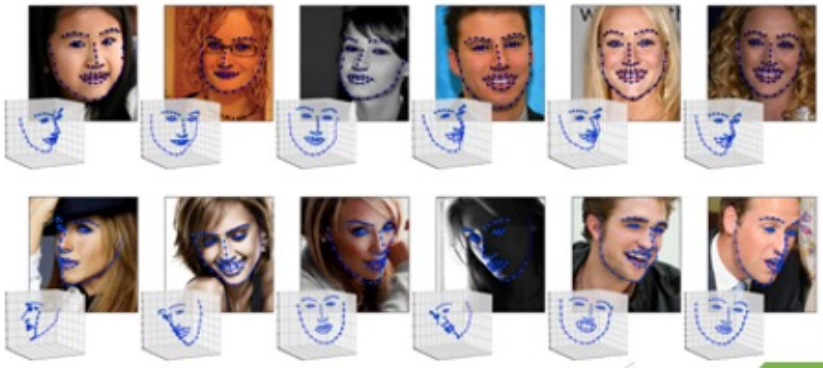
Аналіз методів визначення частоти серцевих здоров'я людини

- Метод заснований на механічних рухах



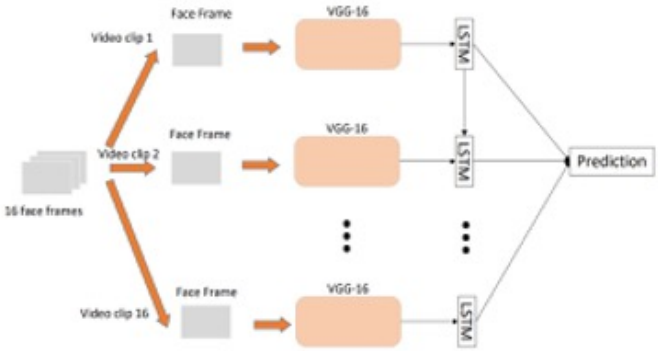
Аналіз методів визначення емоційного стану людини

- Класифікація емоцій за допомогою ключових точок



Аналіз методів визначення емоційного стану людини

- Класифікація емоцій за допомогою методів глибокого навчання.



Класифікація емоцій за допомогою методів глибокого навчання.

Завдання розпізнавання розпізнавання емоцій формулюється так: небхідно створити функцію:

$$F(w) = (F_1(w), F_1(w) \dots F_n(w))$$

де, w - клас зображення

У нашому w випадку класс зображення – це одна з шести базових емоцій людини.

$$F_k(w) = \begin{cases} 1, \text{якщо } w \text{ належить класу} \\ 0, \text{якщо } w \text{ належить класу} \\ \Delta, \text{якщо не відомо} \end{cases}$$

Використані датасети

fer2013

PAB-F

Порівняння датасетів

	«fer2013»	«PAB-F»
Кількість навчальних	28 709	1200
Розмір зображень	320×240	320×240
Колірна гамма	Чорно-білі	Кольорові
Наявність навчених моделей	Є	Немає
Орієнтованість на задачу розпізнавання болю	Немає	Є
Необхідність додаткової логіки після розпвзнавання	Є	Немає

Результати експерименту

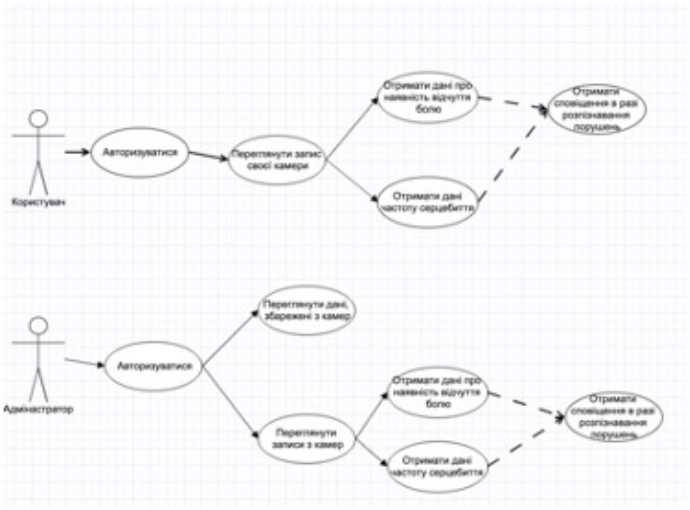
	«fer2013»		«PAB-F»	
	50 епох	100 епох	50 епох	100 епох
Відсок помилкового спрацюванням	18%	16%	15%	13%
Відсок не розпізнання	16%	12%	7%	3%

	«fer2013»		«PAB-F»	
	0,0009	0,005	0,0009	0,005
Відсок помилкового спрацюванням	23%	13%	32%	10%
Відсок не розпізнання	18%	10%	7%	2%

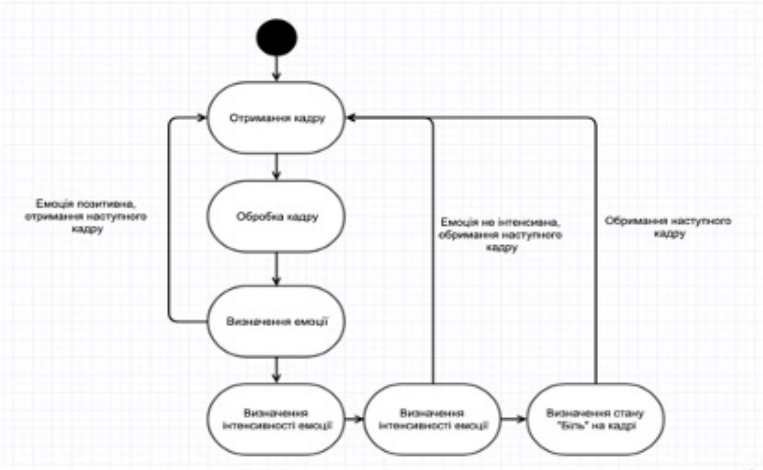
Аналіз методів визначення частоти серцевих здоров'я людини

- Крок 1 - відстеження голови і шиї. Це робиться за допомогою традиційних методик комп'ютерного зору.
- Крок 2 - знаходження вертикальної осі голови. Автори встановили, що вертикальний напрямок найкраще захоплював мимовільний рух через серцебиття, оскільки в русі в горизонтальних напрямках переважало мимовільне колихання.
- Крок 3 - навіть у вертикальному напрямку є багато джерел руху, крім серцевих скорочень. Наприклад, дихання і зміни постави також рухають голову і шию. Щоб видалити ці джерела шуму, автори використовують традиційні методи фільтрації обробки сигналів для орієнтації на рух лише у частотному діапазоні, що відповідає "нормальному" серцебиттю.
- Крок 4 - навіть після фільтрації лише частина вертикального руху голови та шиї пов'язана із частотою серцебиття.

Діаграма прецедентів



Діаграма станів аналізу емоції людини



Діаграма компонентів

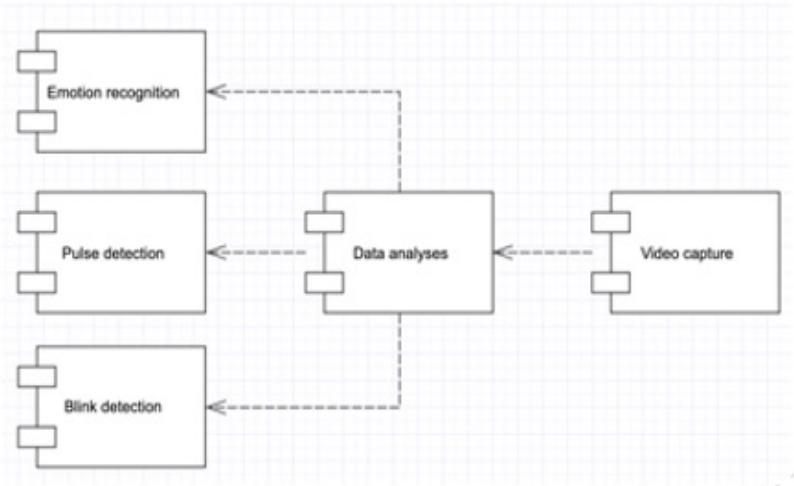
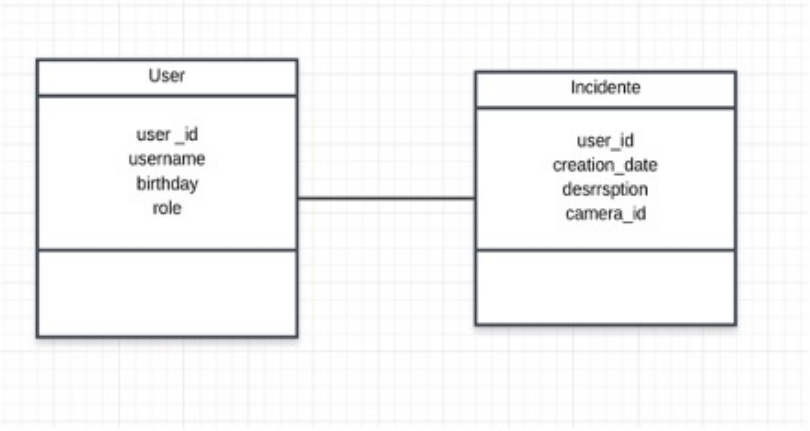
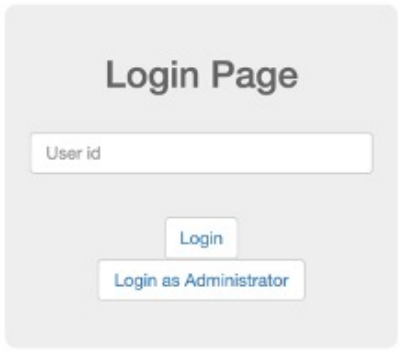


Схема бази даних

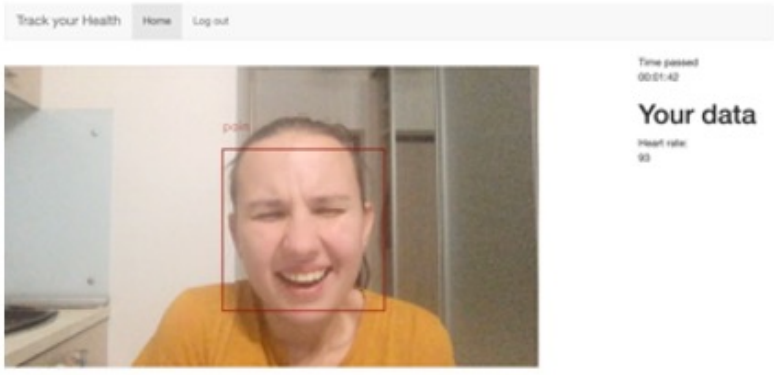




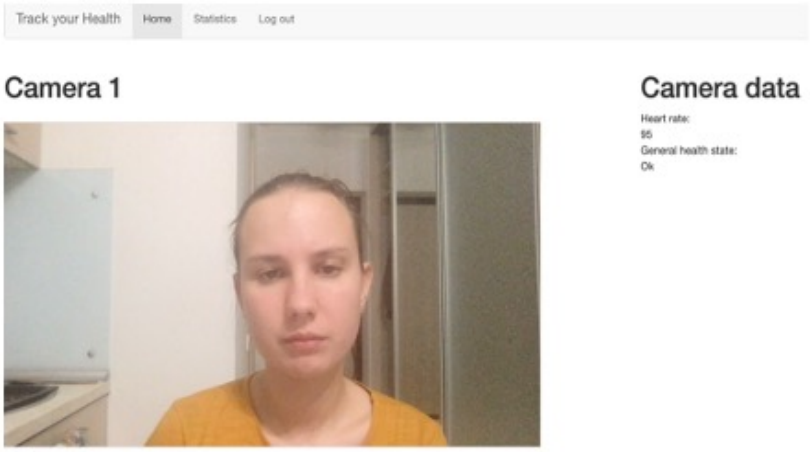
Вхід до системи

- Система не розрахована на реєстрування, адже дані про клієнтів спортзалу вже є у їх базі даних і достатньо тільки зробити синхронізацію.

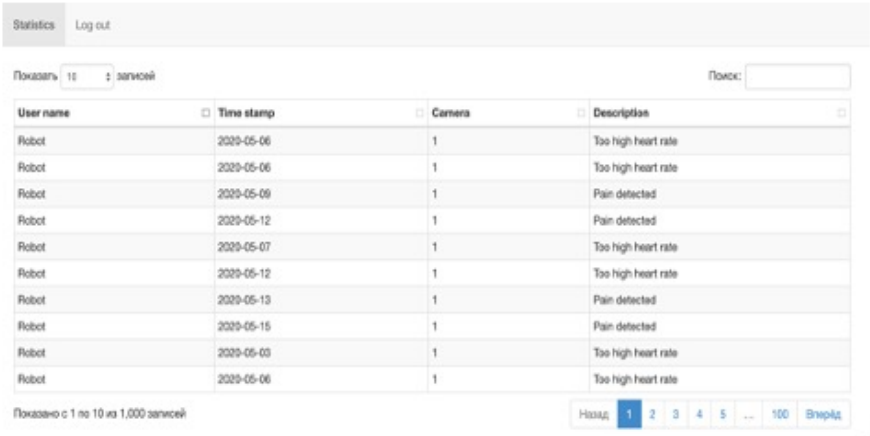
Вікно режиму звичайного користувача



Вікно адміністратора



Вікно перегляду інцидентів



Висновки

Для класифікація емоцій за відео краще використовувати метод гдибокого машинного вавчання.

Датасет для навчання нейронної мережі для визначення болі, рекомендується використовувати спеціалізовані датасети, необхідно вчити нейронну мережу мінімум 100 епох швидкість навчання повинна бути рівна 0,005. За можливості їх треба розширити, хоча б до 1000 зображень для кожного класу.

ДОДАТОК В

Апробація результатів роботи

Білоус Н.В[1], Рассоха О.В[2]

[1]Професор кафедри програмної інженерії, доцент,
Харківський національний університет радіоелектроніки, м. Харків, Україна, nataliya.bilous@nure.ua
[2]Студент кафедри програмної інженерії,
Харківський національний університет радіоелектроніки, м. Харків, Україна, olha.rassokha@nure.ua

ДОСЛІДЖЕННЯ МЕТОДІВ ДЛЯ РОЗРОБКИ ПРОГРАМНОЇ СИСТЕМИ РОЗПІЗНАВАННЯ ЕМОЦІЙ ТА ВИЗНАЧЕННЯ СТАНУ ЗДОРОВ'Я ЛЮДИНИ

Для визначення загального стану здоров’я людини за допомогою розпізнавання емоцій було обрано метод, який базується на машинному навчанні, класифікатор натренований на датасеті «fer2013» та «PAB-F».Було порівняно два способи визначення відчуття болі: визначення відчуття болю за наявністю сильних негативних емоцій та використання для тренування нейронної мережі датасету, з зображеннями людей, поділеними на класи «боляче» - «не боляче». Для датасету «fer2013», щоб визначити наявність болю необхідна після обробка даних. Як правило біль виражається у інтенсивній та тривалій наявності емоції злості та суму. Інтенсивність емоції пропонується вимірювати коефіцієнтом ймовірності визначення емоції, який повертає нейронна мережа. Експериментально визначено, що спеціалізований набір даних краще справляється з поставленою задачею, не дивлячись на те, що має достатньо малу кількість зображень. «fer2013» має високий відсоток помилкового спрацювання. Це можна пояснити тим, що на фотографіях були також явні негативні емоції, які розцінювалися як біль. Кількість епох навчання позитивно впливає на точність нейронної мережі, а збільшення швидкості навчання – негативно. Для перевірки точності роботи нейронних мережі на вхід до модуля перевірки емоційного стану на вхід подавалися зображення знайдені за пошуковим запитом «Гримаса болю» на ресурсі depositphotos.

РОЗПІЗНАВАННІ ЕМОЦІЙ, ОЦІНКА СТАНУ ЗДОРОВ’Я, КОМП’ЮТЕРНИЙ ЗІР, НЕЙРОННІ МЕРЕЖІ, ДАТАСЕТ, FER2013, PAB-F

Для определения общего состояния здоровья человека с помощью распознавания эмоций был выбран метод, основанный на машинном обучении, классификатор натренирован на датасетах «fer2013» и «PAB-F». Было сравнительно два способа определения ощущение боли: определение ощущение боли при наличии сильных негативных эмоций и использования для тренировки нейронной сети датасета с изображениями людей, разделенными на классы «больно» - «не больно». Для датасета «fer2013», чтобы определить наличие боли необходима после обработка данных. Как правило боль выражается в интенсивной и длительной наличии емоциии злости и печали. Интенсивность эмоции предлагаю измерять коэффициентом вероятности определения эмоции, возвращающий нейронная сеть. Экспериментально установлено, что специализированный набор данных лучше справляется с поставленной задачей, несмотря на то, что имеет достаточно малое количество изображений. «Fer2013» имеет высокий процент ложного срабатывания. Это можно объяснить тем, что на фотографиях были также явные негативные эмоции, которые расценивались как боль. Количество эпох обучения положительно влияет на точность нейронной сети, а увеличение скорости обучения - отрицательно. Для проверки точности работы нейронных сети на вход к модулю проверки эмоционального состояния на вход подавались изображения найдены по поисковому запросу «Гримаса боли» на ресурсе depositphotos.

РАСПОЗНАВАНИЕ ЭМОЦИЙ, ОЦЕНКА СОСТОЯНИЯ ЗДОРОВЬЯ, КОМПЬЮТЕРНОЕ ЗРЕНИЕ, НЕЙРОННЫЕ СЕТИ, ДАТАСЕТА, FER2013, PAB-F

To determine the general state of human health using emotion recognition, a method based on machine learning was chosen, the classifier is trained on the dataset "fer2013" and "PAB-F". negative emotions and the use of a dataset to train the neural network, with images of people divided into classes "painful" - "not painful". For the "fer2013" dataset, it is necessary to determine the presence of pain after data processing. As a rule, pain is expressed in the intense and prolonged presence of emotions of anger and sadness. I suggest measuring the intensity of emotion by the probability factor of determining the emotion, which returns the neural network. It has been experimentally determined that a specialized data set better copes with the task, despite the fact that it has a sufficiently small number of images. "Fer2013" has a high percentage of false positives. This can be explained by the fact that the photos also showed obvious negative emotions, which were regarded as pain. The number of learning epochs has a positive effect on the accuracy of the neural network, and the increase in learning speed has a negative effect. To check the accuracy of the neural network at the entrance to the module to check the emotional state at the entrance were images found by the search query "Pain grimace" on the resource depositphotos.

EMOTION RECOGNITION, HEALTH ASSESSMENT, COMPUTER VISION, NEURAL NETWORKS, DATASET, FER2013, PAB-F

Вступ

Розповсюдження камер через мільярди споживчих пристроїв привело до різкого збільшення числа записаних відео, завантажених в соціальних мережах і веб-сайтах обміну відео. Це в свою чергу, стимулювало інтерес в області афективних обчислень, а також машинного навчання і комп'ютерного зору в наукових співтовариствах тож зараз існує багато підходів та алгоритмів аналізу відео та фото матеріалів. Мета роботи – дослідити методи для розробки програмної системи розпізнавання емоцій та визначення стану здоров'я людини. Впровадження нових інформаційних технологій у медицину породжує перехід від

традиційних медичних інформаційно-пошукові системи в інтелектуальних комп'ютерних системах для підтримки прийняття рішень в медицині з передовим математичним апаратом та елементами експертних систем, здатних контролювати діагностичні критерії, побудова передових комп'ютерних діагнозів, впровадження та управління процесом лікування [1].

Об'єкт дослідження – розпізнавання емоцій та визначення стану здоров'я людини. Предмет дослідження – методи для розробки програмної системи розпізнавання емоцій та визначення стану здоров'я людини. Протягом декількох років дослідники критикують поняття, що міміка може легко показати почуття людини. Група вчених з Асоціації психологічних наук провела два роки, вивчаючи більше тисячі наукових праць з розпізнавання емоцій. Вони були в основному зосереджені на тому, як вираз обличчя людини змінюється, коли вона відчуває певні емоції, і як вони трактуються іншими людьми.

Автоматизовані системи (АС) для підтримки прийняття рішень у медичній діагностиці зумовлені низкою причин. Однією з таких причин є необхідність швидкої діагностики в рекреаційній медицині, наприклад, при контролі спортивних навантажень із використанням сучасних мобільних додатків. У таких випадках виникає ситуація, коли для вирішення питання зменшення навантаження або коригування стану пацієнта за допомогою медичних методів потрібна висока працездатність та надійність [2].

Методика дистанційного моніторингу емоцій та стану здоров'я людини за допомогою звичайних камер матиме багато потенційних застосувань. Зміни кольору шкіри, викликані серцевим пульсом, можуть бути зняті звичайними камерами. Зміни викликані болем явно можна побачити на обличчі. Тож пропонується порівняти різні датасети для визначення емоцій, зокрема того, чи відчуває людина біль.

1. Аналіз методів розпізнавання емоцій для визначення стану здоров'я

Найпростіший спосіб визначити емоцію людини за обличчям ґрунтується на класифікації ключових точок (facial landmarks), координати яких можна отримати за допомогою різних алгоритмів PDM, CML, AAM, DPM або CNN. Зазвичай відзначають від 5 до 68 точок, прив'язуючи їх до положення брів, очей, губ, носа, підборіддя, що дозволяє частково захоплювати міміку. Нормовані ключові точки координат можна безпосередньо подати класифікатор (наприклад, SVM або Random Forest) і отримати основне рішення [3].

Просте використання координат без візуальних компонентів призводить до значної втрати корисної інформації, тому для поліпшення системи в цих точках Обчислюються різні дескриптори: LBP, HOG, SIFT, LATCH. Принцип роботи алгоритму показано на рисунку 1.

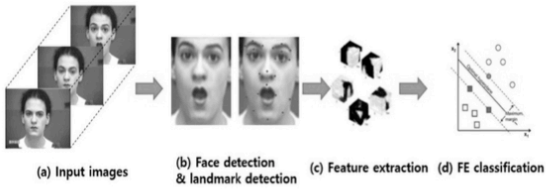


Рис. 1. Принцип роботи алгоритму розпізнавання ключових точок

У своїй статті американські вчені досліджували різні зображення обличчя, отримані від AAM для виявлення болю на обличчі. Вони дослідили два важливі питання щодо автоматичного виявлення болю. По-перше, як слід зобразити обличчя, враховуючи, що доступна нежорстка реєстрація обличчя? По-друге, на якому рівні (тобто на основі послідовностей або кадрів) слід мати один набір даних міток для вивчення автоматичного детектора болю?

Щодо першого питання науковці продемонстрували, що значну користь можна отримати від нежорстких, а не жорстких реєстрацій обличчя. Зокрема, вони продемонстрували, що роз'єднання обличчя на окремі нежорсткі компоненти форми та зовнішнього вигляду забезпечує значне поліпшення продуктивності порівняно з тими, які просто нормалізуються для жорсткої зміни зовнішності (наприклад, просто розміщення очей та нормалізація для перекладу, обертання та масштабу) . Цей результат є вагомим, оскільки в більшості провідних методик блоку дій та болю виявлення застосовують жорсткі, а не нежорсткі реєстрації обличчя [4].

Попередні дослідження з відповідями на фільтри Габора, наприклад, використовували жорстку реєстрацію. Хоча жорстка реєстрація може бути адекватною для деяких застосувань (наприклад, поведеної в поведінці або спонтанної поведінки з невеликим рухом голови поза площиною), для інших вона виявляється, що це не так. Інші дослідження виявили, що жорсткі реєстрації зовнішнього вигляду мали невелику інформаційну цінність у відео з клінічних оцінок болю.

Нежорстке оформлення зовнішнього вигляду значно покращило продуктивність класифікатора. Результати свідчать, що тип реєстрації (жорстка проти нежорстка) може впливати на інформаційну цінність та надійність особливостей зовнішності. Оцінюючи особливості, важливо враховувати питання позачергової ротації та типи реєстрації [5].

Littlewort спочатку виявляв одиниці дії, а потім використовував (передбачувані) дії в класифікаторі для виявлення болю. Дослідження на цю тему тільки починаються.

Більшість попередніх досліджень з виявлення чи розпізнавання виразів обмежувались поставленою поведінкою та описом міміки (наприклад, підрозділи дії чи вирази, що стосуються емоцій, наприклад, щасливі

чи сумні). Наука лише починає вирішувати більш складне питання виявлення суб'єктивних станів, таких як клінічний або викликаний біль.

Людським спостерігачам важко досягти високого рівня надійності і класифікатори, що навчаються на даних, позначених міткою спостерігачів, впливатимуть на джерело відхилення помилок.

Крім того, в тій мірі, в якій виявляються конкретні відчуття на обличчі, класифікатори другого порядку можуть мати перевагу.

Підводячи підсумок, у дослідженні клінічного виявлення болю дослідники виявили, що поєднання не жорстко зареєстрованого вигляду та подібності нормалізованої форми максимізувало виявлення болю як на рівні послідовності, так і на рівні кадру. Навпаки, жорстко зареєстрований вигляд мав мало значення при виявленні болю на рівні послідовності чи кадрів [6].

Ці результати мають наслідки для виявлення болю та загального машинного навчання. Оскільки маркування на рівні послідовності забезпечує збір більших наборів даних, майбутня робота може розглянути гібридні стратегії, що поєднують мітки на рівні послідовностей та кадрів для подальшого покращення виявлення болю та емоція. Сучасні результати досліджень у клінічних болях свідчать про можливість автоматичного виявлення болю в медичних умовах.

Для створення класифікатору, достатньо взяти будь-яку нейронну мережу з базовою архітектурою, попередньо навчену на великому наборі даних емоцій, і перетренувати останні кілька шарів. Таким чином, можливо отримати гарне основне рішення для класифікації емоцій [7]. Але, знімки не точно відображають справжні емоції , що людина переживає в цій ситуації. Тому для підвищення точності необхідно аналізувати послідовність кадрів. Є два способи вирішення цієї задачі.

Перший спосіб аналізу послідовності кадрів полягає в тому, що високо-рівневі ознаки, які класифікують кожен кадр подаються в рекурентну нейронну мережу (наприклад, LSTM) для захвату короткострокової складової. Приклад роботи цього способу показаний на рисунку 2.

Другий спосіб полягає в безпосередній передачі послідовності кадрів, взятих з відео з деяким кроком, до 3D-CNN. Такі нейронні мережі, як CNN використовують згортки з трьома ступенями свободи, які перетворюють чотиривимірні карти у тривимірні карти ознак. Алгоритм цього способу показаний на рисунку 3 [8].

У мімічних реакціях кожної людини є певний набір стандартних параметрів які ділиться на дві категорії: геометричний і поведінковий. Для опису кількісних і якісних параметрів людини використовують систему кодування лицьових рухів. Відео потік даних є послідовним набором кадрів. Метою розпізнавання є об'єднання зображень обличь у в класи, які не перетинаються [8].

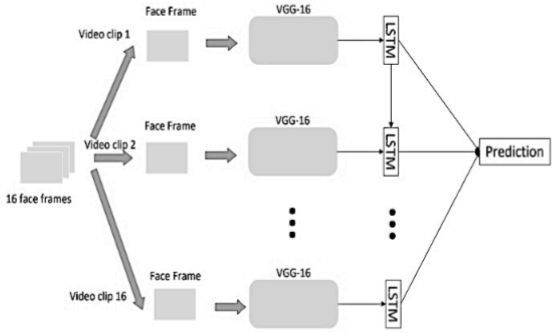


Рис 2. Класифікація ознак за допомогою рекурентної нейронної мережі

Однією із проблем такого методу раніше була недостатня кількість даних для тестового набору, але зараз з'явилась велика кількість наборів різного контексту та призначення. Але зараз деякі дослідження стверджують, що алгоритми простого «зчитування» внутрішніх станів людей лише з аналізу їхніх рухів обличчя, без урахування різних аспектів контексту, не можуть буди повністю достовірними.

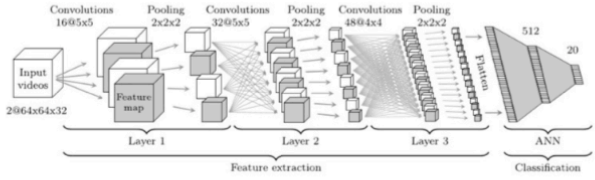


Рис. 3. Аналіз послідовності кадрів з використанням 3D-CNN

Пошук розв'язку виконується за допомогою штучних нейронних мереж. Інваріант – це властивість певного класу (множини) математичних об'єктів, які залишаються однаковими під час певного типу перетворення. Інваріантні моменти – це характерні ознаки, які можуть зустрічатися у кожному зображенні. Найчастіше люди на відеокадрах піддаються різним деформації, характерним для міміки людини [6].

Розв'язання задачі розпізнавання емоцій вважається задачею класифікації, тобто нейрона мережа повинна співставити отриманий набір даних з емоцією, яка відповідає заданому набору параметрів. Вся вибірка поділена на дві підмножини, які не пересікаються: навчальну та тестову. Після навчання штучної нейронної мережі перевіряється якість її навчання на тестовій множені. Нейронна мережа повинна бути настроєна так, щоб при подачі вектору ознак, який відноситься до класу А мережа видавала значення «1» на виході класу А, а на всіх інших виходах «0». Це досягається настройкою мережі методом зворотного поширення помилки: ініціалізація ваги зав'язків: невеликі, випадкові значення та максимальна середньоквадратична помилка; подавання на вхід

нейронної мережі вертор; розповсюдити сигнали до прямих зав'язків; обчислення похибок і похибки вихідного шару нейронів; розрахунок похибки внутрішнього шару нейронів; оновлення ваги зв'язку кожного шару. Алгоритм навчання нейронних мереж: елементи навчальної послідовності подаються на вхід до мережі по одному; елементи тренувальної послідовності порівнювалися з цільовими (бажаними) значеннями; обчислюється функціонал помилки; начення функціоналу та його градієнту використовуються для регулювання вагів. Операції повторюються, доки не виконається певна кількість циклів або доки помилка не зменшиться [7].

У ході дослідження було використано датасети «fer2013» та «PAB-F». Датасет «fer2013» складається з 48x48 піксельних зображень у градаціях сірого. Обличчя були автоматично розміщені так, щоб бути більш-менш по центру і займати приблизно однаково місця в кожному зображенні.

Завдання полягає в категоризації кожної фотографії на основі емоцій, показаних у виразі обличчя в один з семи категорій (0 = злість, 1 = огида, 2 = страх, 3 = щастя, 4 = сум, 5 = здивування, 6 = нейтральні). CSV містить два стовпці, «емоція» та «пікселі». Стовпець «емоція» містить числовий код, що варіюється від 0 до 6 включно, відповідно емоції, яка присутня на зображенні. Стовпець «пікселі» містить рядок, оточений лапками для кожного зображення. Вміст цього рядка – значення пікселів, розділені пробілами. Test. CSV містить лише стовпець «пікселі», і завдання полягає в тому, щоб передбачити емоцію колонки. Навчальний комплект складається з 28 709 прикладів. Публічний набір тестів, складається з 3 589 прикладів [8].

«PAB-F» датасет складається з 320×240 кольорових зображень. Як і у «fer2013» обличчя розміщені так більш-менш по центру, але вони займають різну площу зображення. Завдання полягає в категоризації кожної особи на основі на основі виразу обличчя у одну із таких категорій (0 = не боляче, 1 = дискомфортно, 2 = середній біль, 3 = сильний біль, 4 = дуже сильний біль). Датасет – категоризовані зображення людини розділені по директоріям. Навчальний комплект складається з 1200 прикладів. Набір тестів, складається з 50 прикладів [9].

Для визначення стану здоров'я достатньо тільки негативного спектру емоцій, тобто другої та третьої чверті графіку повного спектру емоцій, показаного на рисунку 4. Якщо емоція знаходиться у першій або четверті чверті, вона не має впливати на результат. Навряд чи можна зберегти гарний настрій, якщо ви постійно відчуваєте біль різного ступеня інтенсивності. Але навіть якщо ви приймаєте ліки, ви все ще можете бути схильні до перепадів. Біль може виражатися такими емоціями, як злість та сум. При цьому емоція дуже інтенсивна [10].

Психолог Miris Kunz університету Монреаля досліджував взаємозв'язок між болем і виразом обличчя. Протестовані двадцять жінок і двадцять чоловіків у віці від 18 до 30 років. На ногах піддослідних були пов'язки, які можуть виробляти тепло. Коли піднялася температура, добровольці, які брали участь в експерименті, продемонстрували ступінь ступінь дискомфорту, а вираз їхніх обличь був сфотографований.

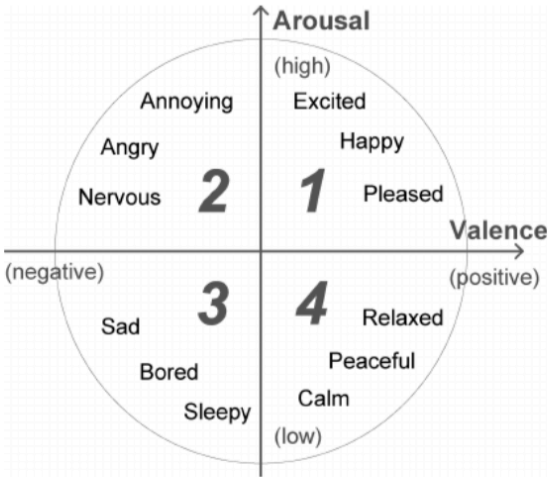


Рис 4. Повний спектр емоцій

Звичайно, Kunz в його дослідженні брав до уваги те, що кожна людина має свій рівень больового порогу. Експеримент показав, що больовий поріг безпосередньо впливає на скорочення лицьових м'язів. Іншими словами, чим інтенсивніша гримаса тим вище степінь відчуття болю [10].

2. Результати досліджень

Аналіз емоцій відбувається наступним чином, спочатку отриманий кадр обробляється та фільтрується, потім фільтровані данні аналізуються нейронною мережею. Нейронна мережа повертає емоцію та її ймовірність, чим інтенсивнішу емоцію відчуває людина, тим більше ймовірність. Позитивні емоції відкидаються одразу. Для відчуття болю характерні сильні, інтенсивні емоції. Алгоритм визначенея відчуття болю через емоції показано на діаграмі станів на рисунку 5. Діаграма станів – орієнтований граф для кінцевого автомата, в якому: вершини позначають стани, а дуги показують переходи між двома станами.

У кожного з поданих датасетів є свої переваги і недоліки: «fer2013» має значно більший навчальний комплект, а «PAB-F» більше підходить тематично. Порівняння датасетів показано на таблиці 1.

Здоров'я людини в момент часу можна поділити на дві категорії: боляче і не боляче. Тоді, для датасету «PAB-F» будемо вважати, що всі типи болю, крім легкого – це боляче, а «не боляче» та «дискомфорт» – не боляче.

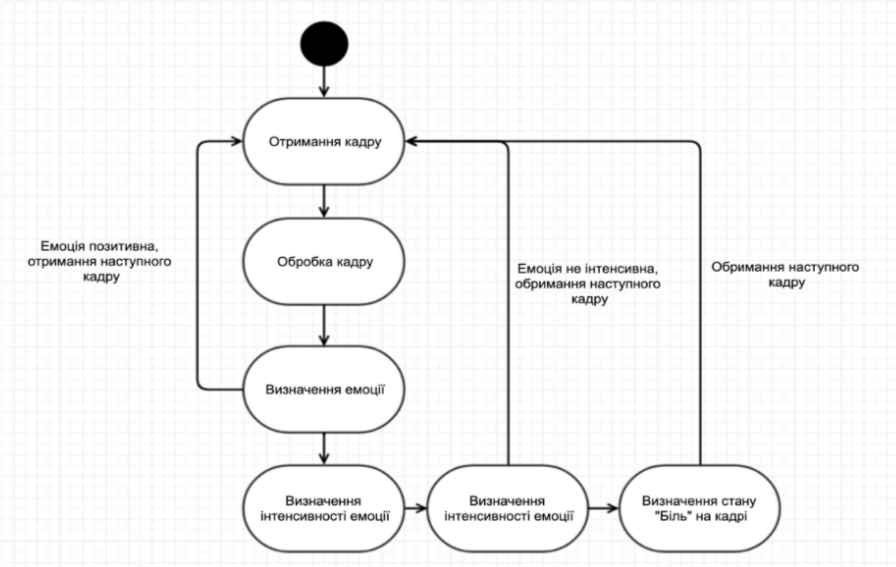


Рис 5. Діаграма станів аналізу емоції людини

Для датасету «fer2013», щоб визначити наявність болю необхідна після обробка даних. Як правило біль виражається у інтенсивній та тривалій наявності емоції злості та суму. Інтенсивність емоції пропоную вимірювати коефіцієнтом ймовірності визначення емоції, який повертає нейронна мережа. Для тренування моделей була використана Google teachablemachine.

Для навчання моделі є кілька налаштувань: епохи, одна епоха - це коли цілий набір даних передається вперед та назад через нейронну мережу лише один раз. швидкість навчання, контролює, наскільки швидко чи повільно модель нейронної мережі навчається.

Таблиця 1.

Порівняння датасетів

	«fer2013»	«PAB-F»
Кількість навчальних	28 709	1200
Розмір тренувальних зображень	48×48	320×240
Колірна гамма	Чорно-білі	Кольорові
Наявність навчання моделей	Є	Є
Орієнтованість на задачу розпізнавання болю	Немає	Є
Необхідність додаткової логіки після розпізнавання	Є	Немає

Для перевірки точності роботи нейронних мережі на вхід до модуля перевірки емоційного стану на вхід подавалися зображення знайдені за пошуковим запитом «Гримаса болю» на ресурсі depositphotos. Усього 130 зображень. Для перевірки також використовувалися зображення за запитом «Емоція люди». Всі зображення цієї категорії однотипні, 150 зображень.

Кожен клас зображень був покладений у окрему директорію. Алгоритм тестування обирав директорію випадковим чином та брав зображення випадкове зображення та подав на вхід модулю розпізнавання емоцій. Коли модуль повертав результат, він порівнювався з назвою директорії, з якої було взяте зображення. Таким чином було проведено 400 тестів та отримано результати подані на таблиці 2.

Таблиця 2.

Результати експерименту з порівняння різної кількості епох навчання.

	«fer2013»		«PAB-F»	
	50 епох	100 епох	50 епох	100 епох
Відсок помилкових спрацювань	18%	16%	15%	13%
Відсок не розпізнання	16%	12%	7%	3%

Тож, експериментально визначено, що спеціалізований набір даних набагато краще справляється з поставленою задачею, не дивлячись на те, що має достатньо малий набір даних. «fer2013» має високий відсоток помилкового спрацювання. Це можна пояснити тим, що на фотографіях були також явні негативні емоції, які розцінювалися як біль. При збільшенні кількості епох навчання збільшується точність та відсоток розпізнання болю, але відсоток помилкового спрацювання майже не змінився. Крім цього пропонується порівняти різні налаштування змін швидкості навчання моделі нейронної мережі. Як зрозуміло з першого експерименту, збільшення епох навчання має позитивний вплив на точність нейронної мережі. Тому, нехай у другому експерименті ми будемо використовувати 100 епох навчання нейронної мережі. Датасети для навчання – «PAB-F» та «fer2013», як і у

експерименті описаному вище. Результати другого експерименту показано на таблиці 3.

Таблиця 3.

Результати експерименту з порівняння різної кількості епох навчання.

	«fer2013»		«PAB-F»	
	0,0009	0,005	0,0009	0,005
Відсок помилкового спрацювання	23%	13%	32%	10%
Відсок не розпізнання	18%	10%	7%	2%

Отже як датасет для навчання нейронної мережі для визначення болі, рекомендується використовувати спеціалізовані датасети, необхідно вчити нейронну мережу мінімум 100 епох швидкість навчання повинна бути рівна 0,005. За можливості їх треба розширити, хоча б до 1000 зображень для кожного класу.

Висновки

Було порівняно два способи визначення відчуття болі: визначення відчуття болю за наявністю сильних негативних емоцій та використання для тренування нейронної мережі датасету, з зображеннями людей, поділеними на класи «боляче» - «не боляче». Експериментально визначено, що спеціалізований набір даних краще справляється з поставленою задачею, не дивлячись на те, що має достатньо малу кількість зображень. «fer2013» має високий відсоток помилкового спрацювання. Це можна пояснити тим, що на фотографіях були також явні негативні емоції, які розцінювалися як біль. Кількість епох навчання позитивно впливає на точність нейронної мережі, а збільшення швидкості навчання – негативно. Доведено, що відчуття болю у момент часу може досить точно бути визначено за допомогою відеоспостереження.

Література

[1] *Bilous, N and Povoroznyuk, A and Kozina, O* Synthesis of structured models of computer systems in medical diagnosis, 2009 [Електронний ресурс] – Режим доступу: http://openarchive.nure.ua/bitstream/document/6432/1/ijitk03-3-p03.pdf

[2] *Shcherbakova G., Krylov V., Bilous N.*, Methods of automated classification based on wavelet-transform for automated medical diagnostics, Information Technologies in Innovation Business Conference (ITIB) 2015, pp. 7-10. DOI: 10.1109/ITIB.2015.7355048

[3] *Dornaika, F.; Raducanu, B.* Efficient facial expression recognition for human robot interaction. In Proceedings of the 9th International Work-Conference on Artificial Neural Networks on Computational and Ambient Intelligence, San Sebastián, Spain, 20–22 June 2007; pp. 700–708.

[4] *Bartneck, C.; Lyons, M.J.* HCI and the face: Towards an art of the soluble. In Proceedings of the International Conference on Human-Computer Interaction: Interaction Design and Usability, Beijing, China, 22–27 July 2007; pp. 20–29

[5] *Walecki, R.; Rudovic, O.; Pavlovic, V.; Schuller, B.; Pantic, M.* Deep structured learning for facial action unit intensity estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 3405–3414

[6] *Schlüter S. et al.* Image processing of multiphase images obtained via X-ray microtomography: a review // Water Resources Research. – 2014. –№. 50(4). – P. 3615-3639.

[7] *Yin, L.; Wei, X.; Sun, Y.; Wang, J.; Rosato, M.J.* A 3D facial Expression database for facial behavior research. In Proceedings of the International Conference on Automatic Face and Gesture Recognition, Southampton, UK, 10–12 April 2006; pp. 211–216.

[8] *Jiang, B.; Valstar, M.F.; Pantic, M.* Action unit detection using sparse appearance descriptors in space-time video volumes. In Proceedings of the IEEE International Conference and Workshops on Automatic Face & Gesture Recognition, Santa Barbara, CA, USA, 21–25 March 2011; pp. 314–321

[9] *Hasani, B.; Mahoor, M.H.* Facial expression recognition using enhanced deep 3D convolutional neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Hawaii, HI, USA, 21–26 July 2017; pp. 1–11.

[10] *Kunz, M., Scharmann S., Uli Hemmeter, Stefan B*. The facial expression of pain in patients, Califirnia, October 2009

Надійшла до редколегії 21.05.2020