**МИНОБРНАУКИ РОССИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ**

**УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМ. Р.Е. АЛЕКСЕЕВА»**

**(НГТУ)**

Кафедра «Вычислительные системы и технологии»

Дисциплина «Методы и средства обработки сигналов»

**Отчёт по 2 лабораторной работе**

Выполнила:

Студентка группы 23 – ВМз

Бутыкова Юлия Олеговна

Проверил:

доцент, кандидат технических наук

Авербух Михаил Леонидович

Нижний Новгород, 2025

**ПОСТАНОВКА ЗАДАЧИ**

**Цели работы:**

1. Реализовать алгоритм наложения фильтра F₁ теории активного восприятия (один уровень Q-пирамиды) на часть изображения;
2. Построить график функции *y*(*x*) = *μ*1​(*x*) для 2ух заданных изображений;
3. Реализовать отображение векторов x и y (в виде таблицы). Построение графика y(x) по указанным векторам;
4. Попробовать различные значения w и d и определить соответствуют ли характерные точки на графике (т.е. локальные минимумы или максимумы и т.д.) координатам x, где по мнению студента начинаются или заканчиваются изображения отдельных символов (т.е. левая и правая граница буквы) на исходном изображении;
5. Обобщить результаты экспериментов.

**Данные:**

* 2 изображения;
* Ширина фильтра w задается пользователем и является положительной целой величиной кратной 4;
* Шаг сдвига фильтра d задается пользователем;
* Высота фильтра берётся равной высоте исходного изображения.

**ХОД ВЫПОЛНЕНИЯ – ДЕМОНСТРАЦИЯ РАБОТЫ ПРОГРАММЫ**

**Задачи:**

* + 1. Наложить фильтр F1 на часть исходного изображения вдоль левой границы изображения, высотой, как было сказано, в высоту изображения, и шириной w;
    2. Вычислить y(x0), то есть самое первое значение в векторе y, за x0 берется середина фильтра (т.е. w/2);
    3. Вычислить разницу между суммарной яркостью всех пикселей правой половины этой части изображения и левой половины этой изображения. В результате получаем сигнал y(x), где y – разница яркостей, а x – координата центра части изображения по горизонтали.
    4. Затем область наложения фильтра сдвигается вправо на d, и вычисляется следующее значение вектора y, соответствующее x = w/2 + d;
    5. так продолжать, пока не будет достигнута правая границы исходного изображения.
    6. Сформировать векторы *x* и *y*.
    7. Вывести векторы в виде таблицы.
    8. Построить график *y*(*x*).
    9. Провести эксперименты с разными *w* и *d*.
    10. Проанализировать, соответствуют ли экстремумы графика границам символов.
    11. Сохранить график в формате BMP или SVG (без GUI).

Код программы на языке Python. Файл programm.py

import matplotlib.pyplot as plt

import cv2

import numpy as np

from scipy.signal import find\_peaks

def compute\_mu1(image, w, d):

# Вычисляет mu1(x) — разницу яркостей правой и левой половин фильтра.

    height, width = image.shape

    x\_vals = []

    y\_vals = []

    for x in range(0, width - w + 1, d):

    # Область фильтра: от x до x + w

        roi = image[:, x:x + w]

        half\_w = w // 2

    # Левая и правая половины

        left\_half = roi[:, :half\_w]

        right\_half = roi[:, half\_w:]

    # Суммарная яркость

        left\_sum = np.sum(left\_half)

        right\_sum = np.sum(right\_half)

        try:

            mu1 = np.subtract(right\_sum, left\_sum, dtype=object)

        except OverflowError as e:

            print(f"Overflow error occured: {e}")

        x\_vals.append(x + half\_w)  # координата центра фильтра

        y\_vals.append(mu1)

    return np.array(x\_vals), np.array(y\_vals)

def main():

# Ввод параметров

    w = int(input("Введите ширину фильтра w (кратно 4): "))

    if w <= 0 or w % 4 != 0:

        raise ValueError("w должно быть положительным и кратным 4")

    d = int(input("Введите шаг сдвига d: "))

    if d <= 0:

        raise ValueError("d должно быть положительным")

    image\_paths = input("Введите имя файла: ")

# Загрузка изображения (в градациях серого)

    image = cv2.imread(image\_paths, cv2.IMREAD\_GRAYSCALE)

    if image is None:

        raise FileNotFoundError(f"Не удалось загрузить изображение: {image\_paths}")

    print(f"\nОбработка изображения {image\_paths}:")

    height, width = image.shape

# Вычисление mu1(x)

    x\_vals, y\_vals = compute\_mu1(image, w, d)

# Вывод таблицы

    print("x\t\ty")

    for xi, yi in zip(x\_vals, y\_vals):

        print(f"{xi}\t\t{yi}")

# Поиск экстремумов

    max\_indices = find\_peaks(y\_vals, height=np.mean(y\_vals))[0]

    min\_indices = find\_peaks(-y\_vals, height=-np.mean(y\_vals))[0]

# Визуализация

    fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(14, 10), gridspec\_kw={'height\_ratios': [2, 1]})

# Верхняя панель: изображение с границами

    ax1.imshow(image, cmap='gray')

    ax1.set\_title('Исходное изображение с размеченными границами символов', fontsize=12)

    ax1.set\_xticks([])

    ax1.set\_yticks([])

# Отмечаем все экстремумы вертикальными линиями на изображении

    for x\_peak in x\_vals[max\_indices]:

        ax1.axvline(x\_peak, color='red', linestyle='--', linewidth=2, alpha=0.8)

        ax1.text(x\_peak, height - 10, f'{x\_peak}', color='red', fontsize=9, ha='center', va='top')

    for x\_valley in x\_vals[min\_indices]:

        ax1.axvline(x\_valley, color='green', linestyle='--', linewidth=2, alpha=0.8)

        ax1.text(x\_valley, height - 10, f'{x\_valley}', color='green', fontsize=9, ha='center', va='top')

# Нижняя панель: график y(x)

    ax2.plot(x\_vals, y\_vals, label=r'$y(x) = \mu\_1(x)$', color='blue', linewidth=1.5)

    ax2.set\_xlabel('x (координата центра фильтра)', fontsize=10)

    ax2.set\_ylabel(r'$\mu\_1$', fontsize=10)

    ax2.grid(True, linestyle=':', alpha=0.6)

    ax2.legend(fontsize=9)

# Отмечаем экстремумы точками

    ax2.plot(x\_vals[max\_indices], y\_vals[max\_indices], 'ro', label='Максимумы', markersize=6)

    ax2.plot(x\_vals[min\_indices], y\_vals[min\_indices], 'go', label='Минимумы', markersize=6)

# Вертикальные линии от экстремумов (связывают график и изображение)

    for x\_peak in x\_vals[max\_indices]:

        ax2.axvline(x\_peak, color='red', linestyle='-', linewidth=1.5, alpha=0.7)

    for x\_valley in x\_vals[min\_indices]:

        ax2.axvline(x\_valley, color='green', linestyle='-', linewidth=1.5, alpha=0.7)

# Общий заголовок

    fig.suptitle(f'Сопоставление изображения и графика $y(x)$\n' f'w = {w} px, d = {d} px', fontsize=14, y=0.98)

# Оптимизация отступов

    plt.tight\_layout()

    plt.subplots\_adjust(top=0.92)  # чтобы заголовок не перекрывался

# Сохранение и показ

 output\_path = f"result\_{w}-{d}.svg"

    plt.savefig(output\_path, format='svg', dpi=300, bbox\_inches='tight')

    print(f"График сохранён {w}-{d}")

    plt.show()

    plt.close()

if \_\_name\_\_ == '\_\_main\_\_':

    main()

Сначала протестируем первое изображение 1picture.bmp

Изображение выглядит как Шрифт, типография, текст, Графика

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 1. "1picture.bmp"

Ширину сдвига и шага попробуем 4 варианта: 8-1, 12-2, 16-4, 32-8

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как текст, снимок экрана, черный, черно-белый

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.1 **Вариант w-8; d-1**

Изображение выглядит как текст, снимок экрана, черный, черно-белый

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как текст, снимок экрана, черный, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как текст, снимок экрана, черный, черно-белый

Контент, сгенерированный ИИ, может содержать ошибки.

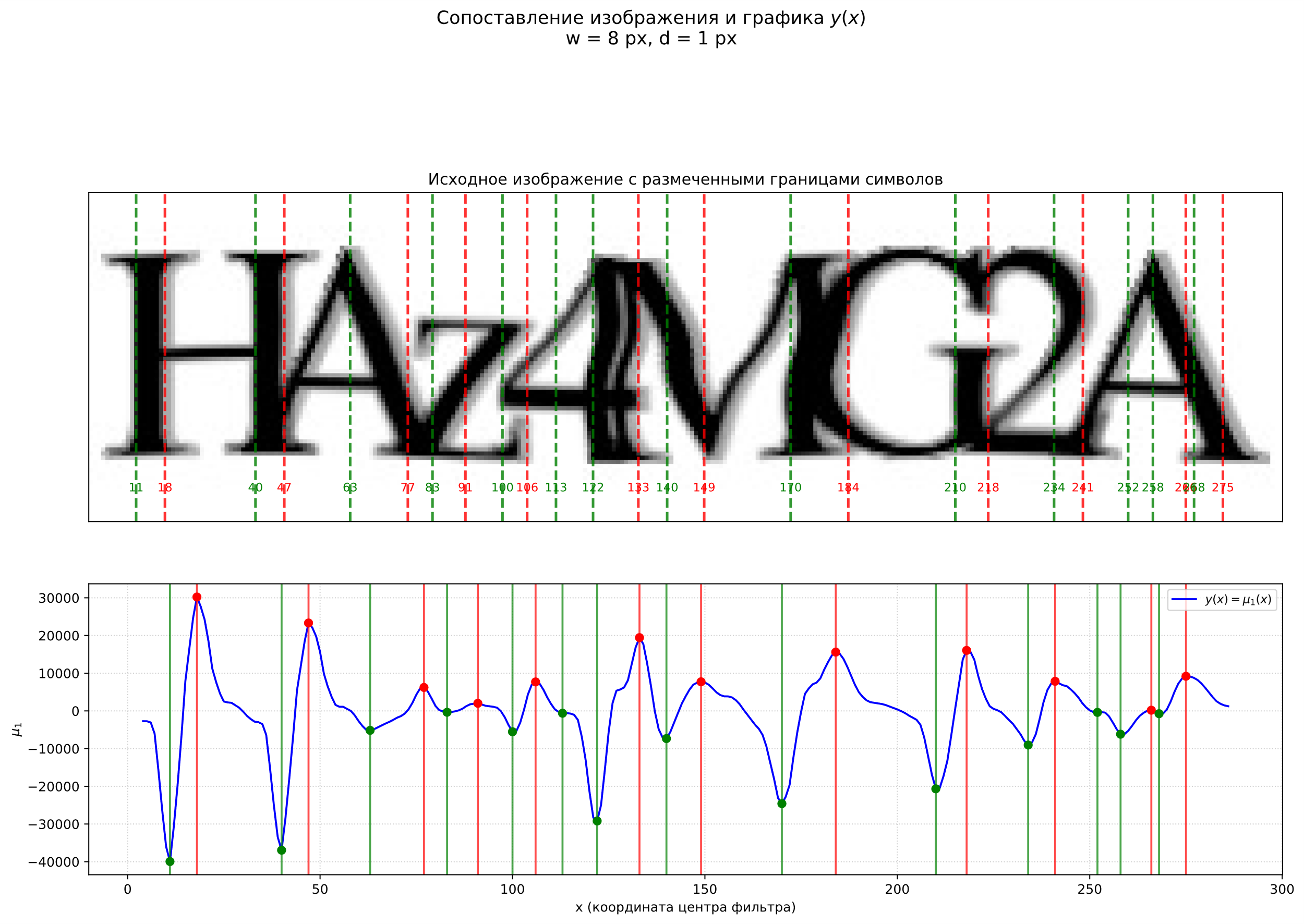


Рисунок 2. Сопоставление графика и рисунка - 8-1

Изображение выглядит как текст, снимок экрана, черный, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как текст, снимок экрана, черный, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.***2 Вариант w-12; d-2***

Изображение выглядит как текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

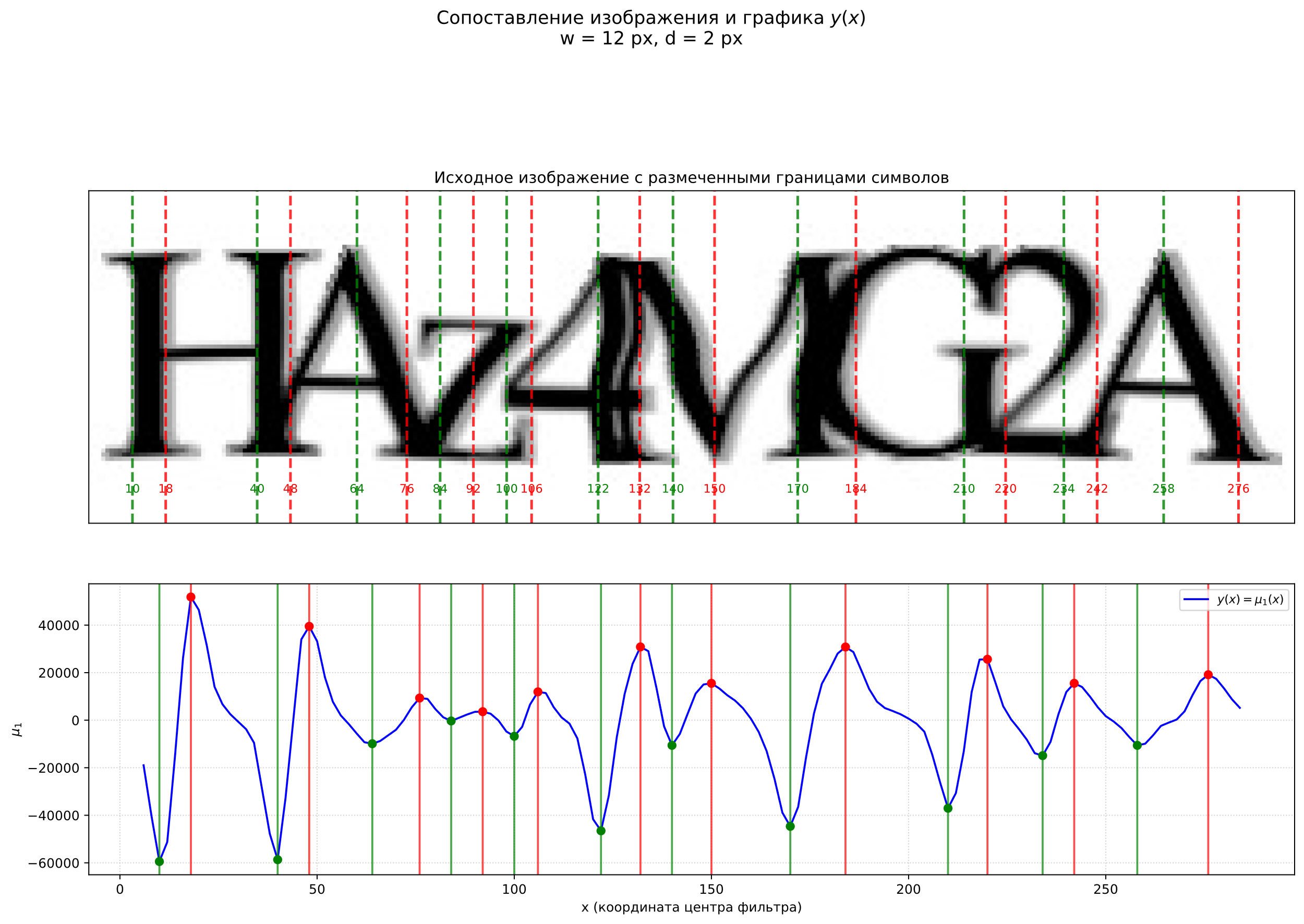


Рисунок 3. Сопоставление графика и рисунка. 12-2

Изображение выглядит как текст, снимок экрана, черно-белый, черный

Контент, сгенерированный ИИ, может содержать ошибки.***3 Вариант w-16; d-4***

Изображение выглядит как текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

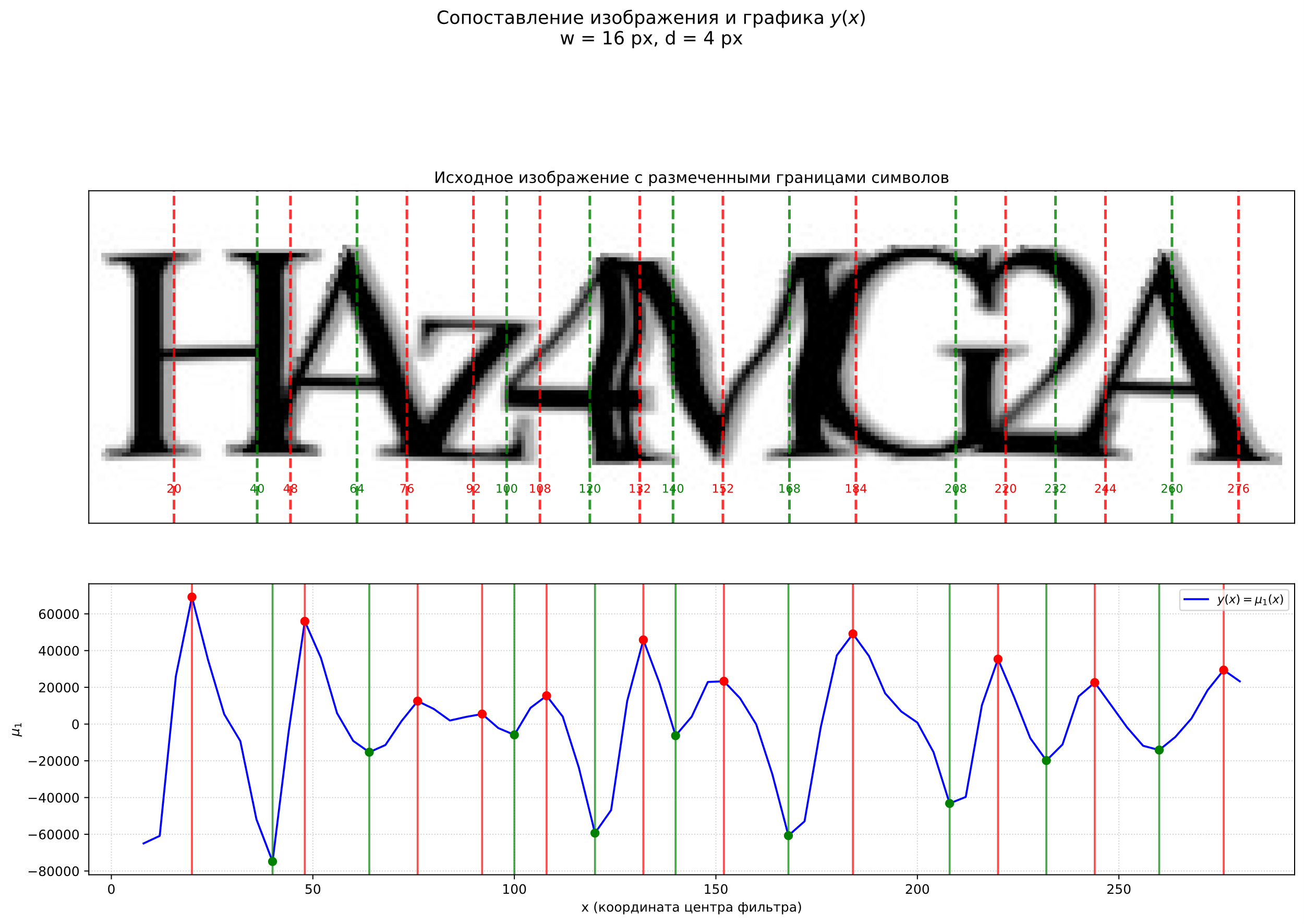


Рисунок 4. Сопоставление графика и рисунка 16-4

***4 Вариант w-32; d-8***

Изображение выглядит как текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

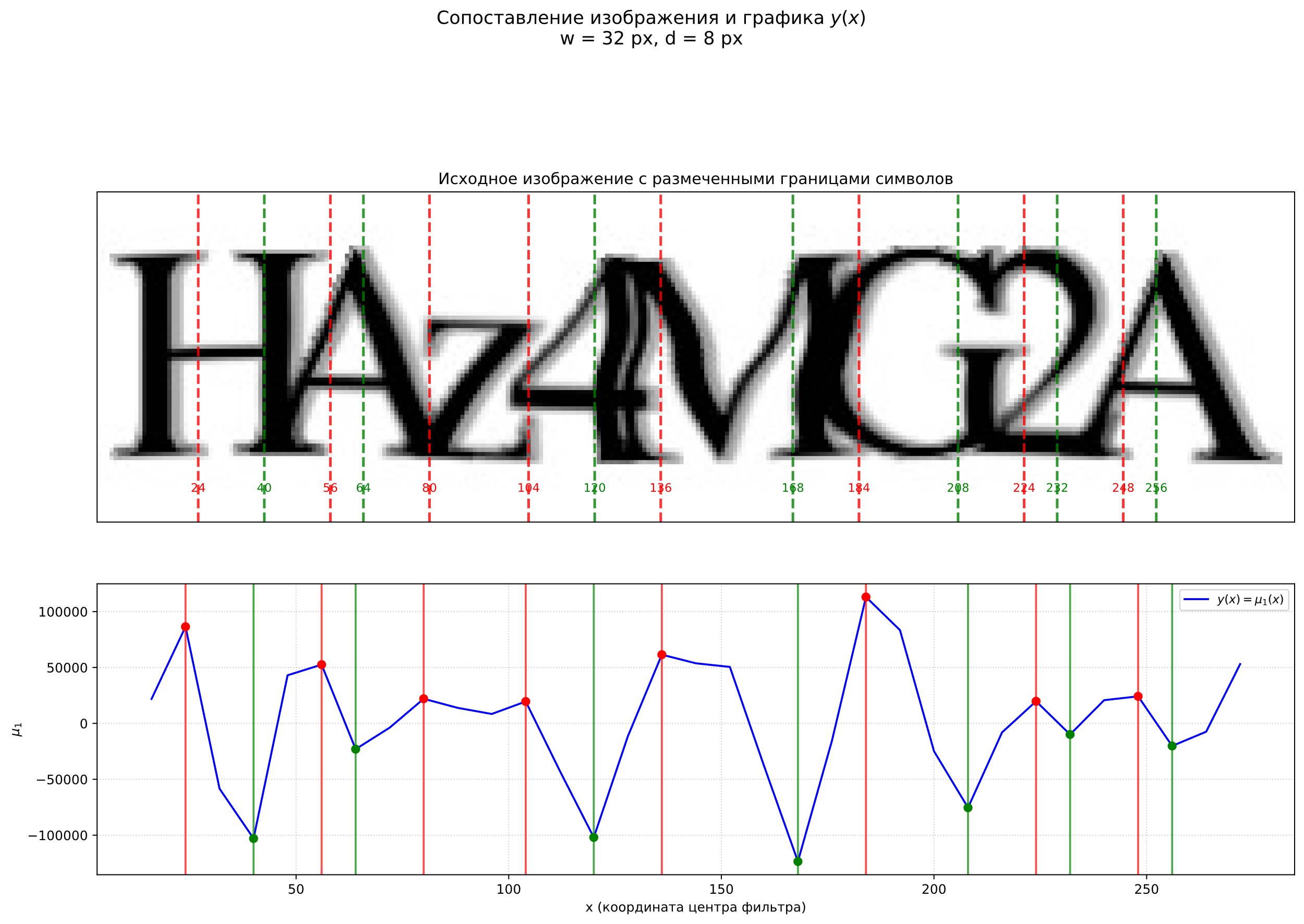


Рисунок 5. Сопоставление графика и рисунка 32-8

**Выводы:**

Программа работает, находит левые и правые границы букв. Но есть некоторые трудности нахождения границ: когда буквы тесно расположены к друг другу, буква z, цифра 2 сложные, волнистые границы как у цифры 4 и буквы М.

Чем больше d и w, тем больше пропусков. А чем меньше w и d тем больше детализация, но тем дольше вычисления и есть лишний шум.

Наиболее оптимальный вариант по моему мнению 12-2.

**Протестируем второе изображение 2picture.bmp**

Изображение выглядит как рукописный текст, Шрифт, каллиграфия, белый

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6. 2picture.bmp

Так же попробуем различные варианты w и d: 8-1, 12-2, 16-4,32-8

Изображение выглядит как текст, снимок экрана, черный, черно-белый

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как снимок экрана, текст, черный

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как снимок экрана, текст, черный

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как снимок экрана, текст, черный, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

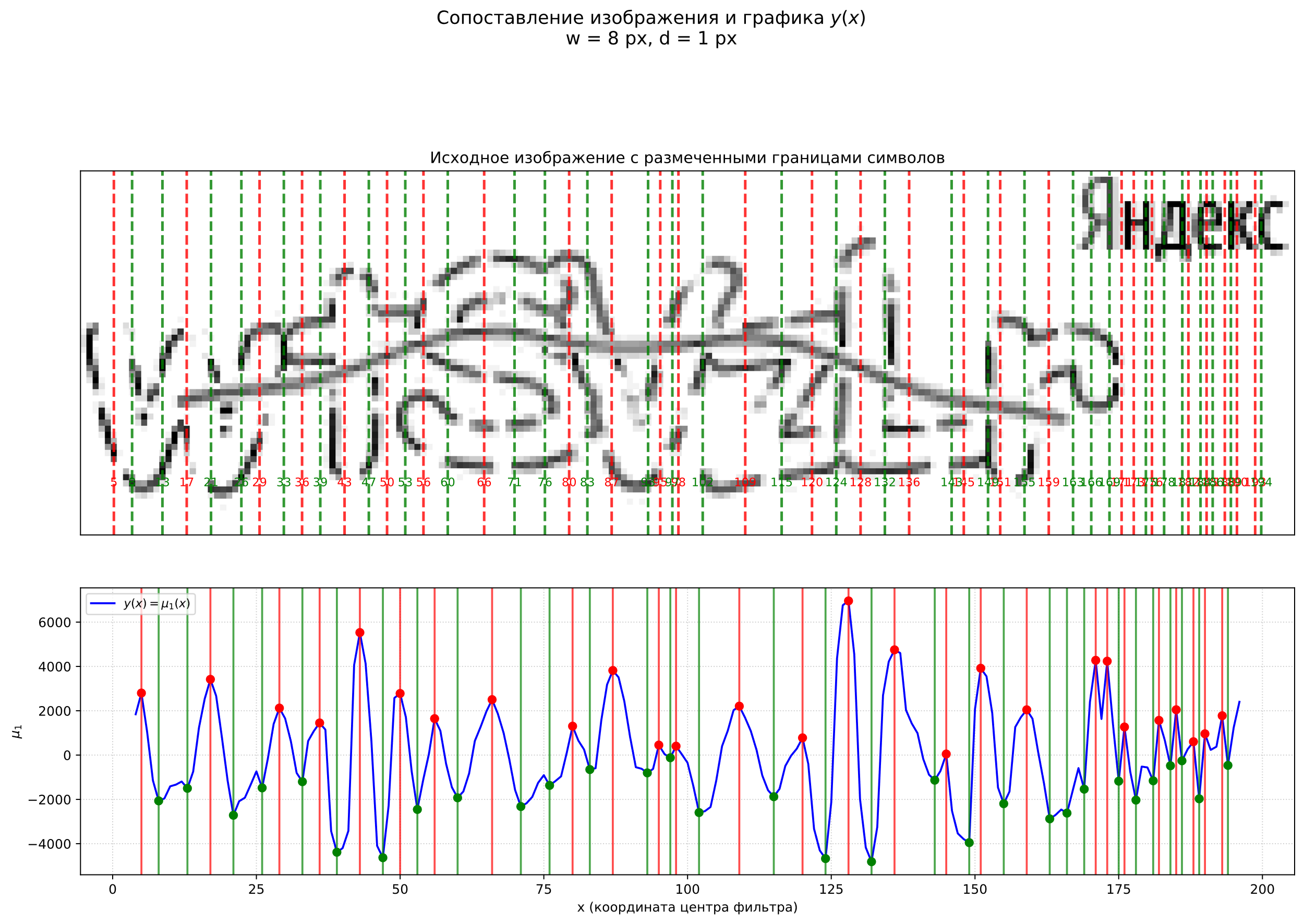


Рисунок 7. Сопоставление графика и рисунка 8-1

*Изображение выглядит как снимок экрана, текст, черный, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.****2 Вариант 2picture.bmp w-12 d-2***

Изображение выглядит как текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

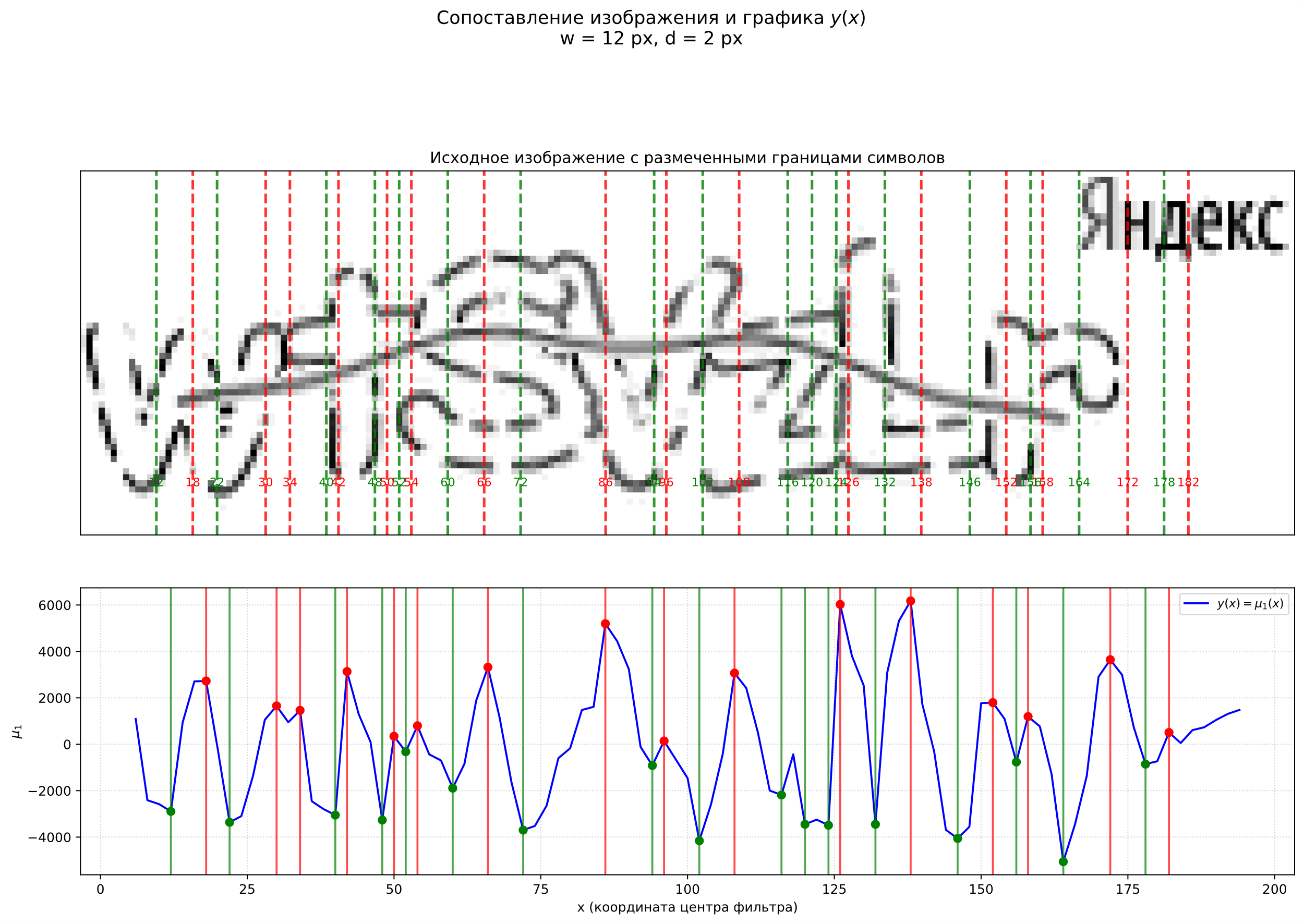


Рисунок 8. Сопоставление графика и рисунка 12-2

***3 Вариант 2picture.bmp w-16 d-4***

Изображение выглядит как текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

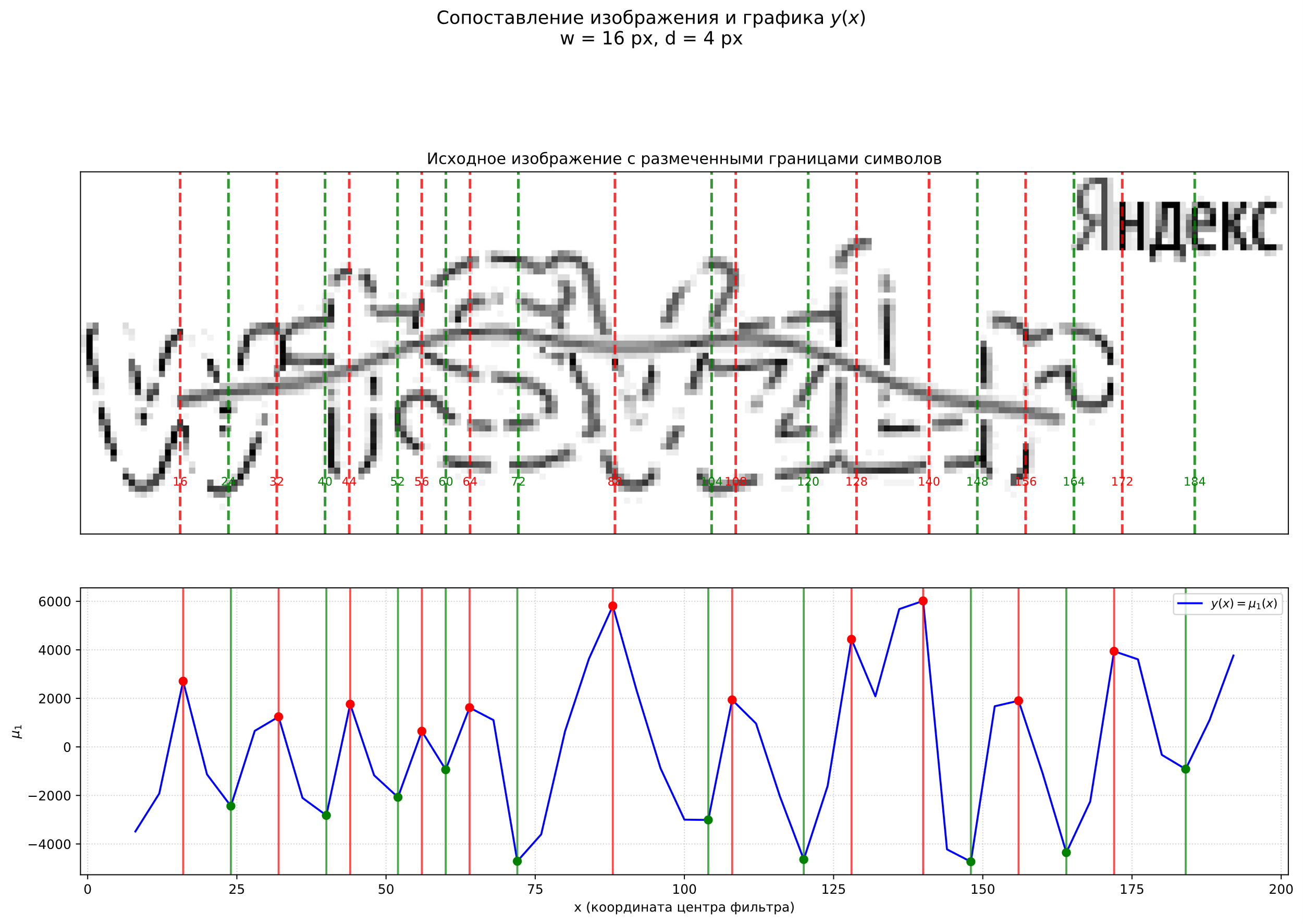


Рисунок 9. Сопоставление графика и рисунка 16-4

***4 Вариант 2picture.bmp 32-8***

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

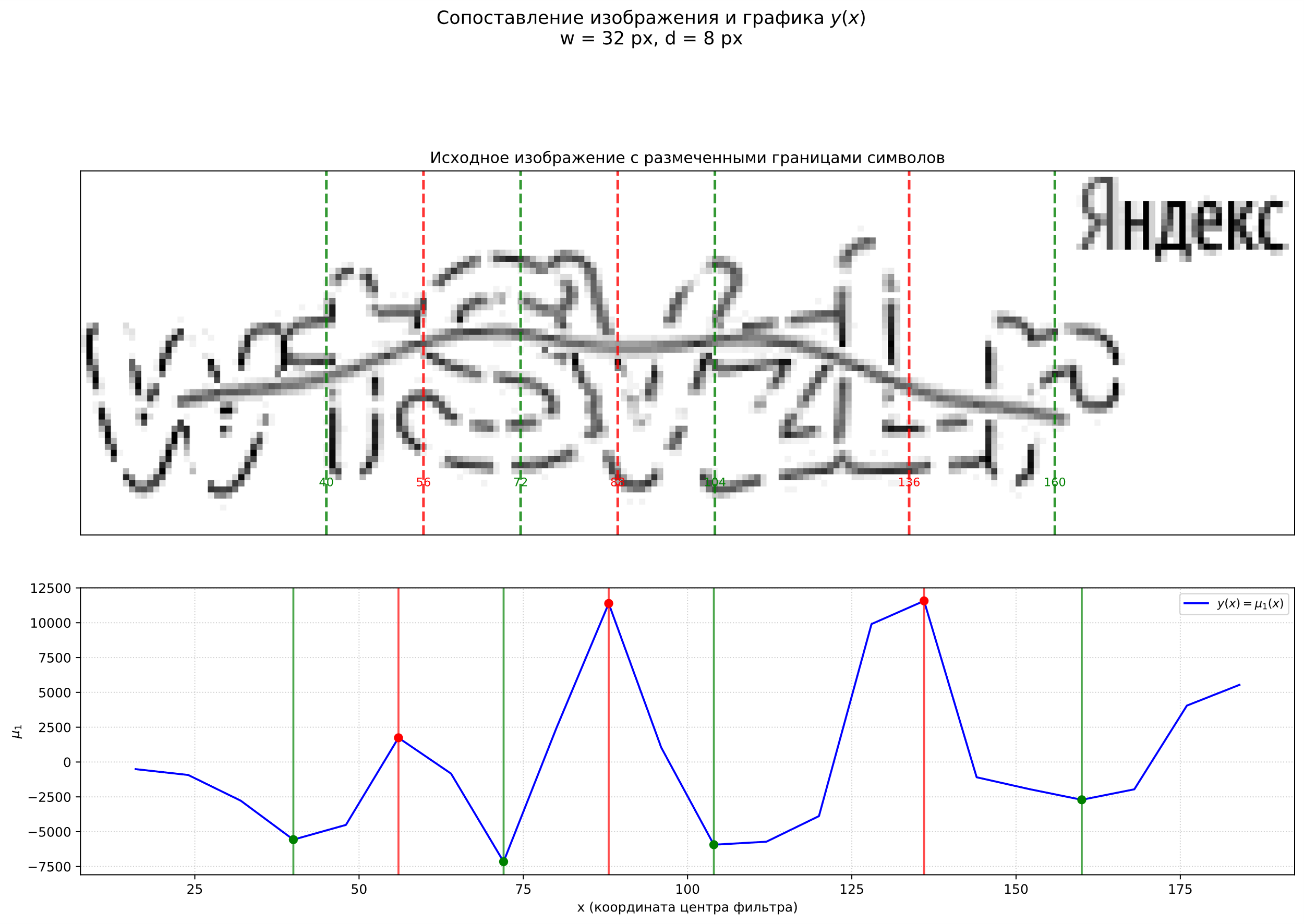


Рисунок 10. Сопоставление графика и рисунка 32-8

**Выводы:**

Данную картинку очень трудно пропускать через фильтр. Мешают размытые границы и волнистая линия в центре. При меньших значениях w и d очень много шума, лишних линий, при больших значениях w и d очень много пропусков.

**Выводы**

Разработана программа, реализующая наложение фильтра F₁ теории активного восприятия на изображение. Программа строит график функции *y*(*x*) = *μ*1​(*x*) и позволяет проанализировать соответствие характерных точек графика границам символов на изображении.

Проанализировав различные показатели w и d сделали выводы, что при увеличении *w*:

* + уменьшается число ложных срабатываний, но растут пропуски узких символов;
  + сглаживается график, экстремумы становятся шире.

При увеличении *d*:

* + ускоряется расчёт, но снижается точность локализации границ;
  + могут пропускаться близкие пары экстремумов.

В первом изображении программа работает лучше, там, где цифры или буквы стоят отдельно, не слитно и границы лучше видно, график наиболее точно показывает левые и правые границы символов.  
Со вторым изображением программа работает хуже. Трудно анализировать картинку, буквы размыты, границы не четкие и плюс есть волнистая линия, которая мешает анализу.