DOI 10.37539/2949-1991.2024.2.13.020
УДК 616-006.04, 616.36, 004.93'1

Полохина Людмила Сергеевна, магистрант,
МИРЭА – Российский технологический университет,
г. Москва

ОБРАБОТКА ИЗОБРАЖЕНИЙ ПЕЧЕНИ НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ PYTHON
LIVER IMAGE PROCESSING IN THE PYTHON PROGRAMMING LANGUAGE

Аннотация: Целью данной работы является улучшение восприятия УЗИ-снимков врачом-специалистом посредством постобработки полученных изображений. Результаты, полученные путем применения методики, изложенной в работе, позволяют наглядно оценить эффективность представленного способа.

Abstract: The purpose of this work is to improve the perception of ultrasound images by a doctor by postprocessing the images obtained. The results obtained by applying the methodology described in the work allow us to visually assess the effectiveness of the presented method.

Ключевые слова: печень, опухоль, Python, фильтрация.

Keywords: liver, tumor, Python, filtration.

При обследовании печени методом УЗИ могут возникать сложности с визуальным определением патологии ввиду слабой контрастности УЗ-изображения данного органа и наличия большого количество проходящих через него кровеносных сосудов. С целью улучшения визуализации было принято решение создать программный код на языке Python [1], который не просто сделает изображение более «читаемым» для специалиста, облегчая восприятие картинки человеческим глазом, но и будет помогать определять паталогический очаг без ручной настройки под каждое отдельное изображение.

Для демонстрации работы используемого кода было взяты анонимные УЗ-снимки печени с различными видами новообразований на разных стадиях.

В первом блоке кода необходимо подключить библиотеки, представленные на рисунке 1, нужные для работы: «opencv-python», предназначенную для решения проблем компьютерного зрения, и математическая библиотека «numpy».

```
!pip install opencv-python
!pip install numpy
```

Рис. 1 – подключение библиотек

Второй блок программы, показанный на рисунке 2, – основной. Он включает в себя все шаги, от загрузки изображения до вывода итогового.

```
import cv2
import numpy as np

# загружаем изображение
my_photo = cv2.imread('./photo/1.jpg')
```

Рис. 2 – загрузка изображения

Далее применяется адаптивный алгоритм повышения контрастности [2], где сперва программа создает адаптивную гистограмму (при этом изображение разбивается на блоки 6x6, для каждого из которых гистограмма строится отдельно), а затем параметр «clipLimit» определяет степень увеличения контрастности изображения. Данный шаг показан на рисунке 3.

```
clahe = cv2.createCLAHE(clipLimit=1100., tileGridSize=(6,6))
```

Рис. 3 – адаптивный алгоритм повышения контрастности

Чтобы наложить на изображение выровненные гистограммы, оно преобразуется в формат «LAB» (представлено на рисунке 4). Он представляет собой 1 канал светлоты L и 2 канала цвета A и B.

```
lab = cv2.cvtColor(my_photo, cv2.COLOR_BGR2LAB)
l, a, b = cv2.split(lab)
```

Рис. 4 – преобразование изображения в формат LAB

Далее необходимо применить гистограммы к каналу светлоты L (рисунок 5), затем эти каналы объединить и выполнить обратное преобразование с помощью «cv2.cvtColor» [3] (трансформирует изображение из одного пространства цвета в другое).

```
# применяем выровненные гистограммы (только к каналу L - светлота)
l2 = clahe.apply(l)

# объединяем эти каналы и производим обратное преобразование
lab = cv2.merge((l2,a,b))
my_photo_2 = cv2.cvtColor(lab, cv2.COLOR_LAB2BGR)
```

Рис. 5 – применение выровненных гистограмм по каналу L,
их объединение и обратное преобразование

Следом применяем медианный фильтр, как показано на рисунке 6.

```
median_image  = cv2.medianBlur(my_photo_2,7)
```

Рис. 6 – применение медианного фильтра

Так как на изображении все еще сохраняются шумы, и оно недостаточно резкое, необходимо применить фильтр Гаусса [4], представленный на рисунке 7.

```
kernel = np.array([[-1,-1,-1], [-1,9,-1], [-1,-1,-1]])
im = cv2.filter2D(median_image, -1, kernel)
```

Рис. 7 – применение Гауссовского фильтра

Полученное изображение необходимо конвертировать в цветовое пространство в оттенках серого, как показано на рисунке 8.

```
img_grey = cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)
```

Рис. 8 – конвертирование изображения

Часть кода, представленная на рисунке 9, помогает отследить изменения изображения по мере выполнения кода.

```
cv2.imshow('origin', img_grey)
cv2.waitKey()
cv2.destroyAllWindows()
```

Рис.9 – вывод получившегося изображения

На выведенном изображении видна опухоль, но мешает большое количество спекл-шума. Поэтому на рисунке 10 представлено действие функции «HoughCircles» [5], в которую уже встроен детектор контуров.

```
rows = img_grey.shape[0]
circles = cv2.HoughCircles(img_grey, cv2.HOUGH_GRADIENT, 1, rows / 8,
                           param1=120, param2=90,
                           minRadius=1, maxRadius=100)
```

Рис. 10 – применение функции HoughCircles

Остается отобразить найденные окружности на исходном изображении (рисунок 11).

```
res = np.zeros(my_photo.shape)
if circles is not None:
    circles = np.uint16(np.around(circles))
    for i in circles[0, :]:
        center = (i[0], i[1])
        # circle center
        cv2.circle(my_photo, center, 1, (0, 100, 100), 3)
        # circle outline
        radius = i[2]
        cv2.circle(my_photo, center, radius, (255, 0, 255), 3)
```

Рис. 11 – отображение найденных окружностей на исходном изображении

Итоговый результат выводим подобным образом (рисунок 12), как было представлено на рисунке 9.

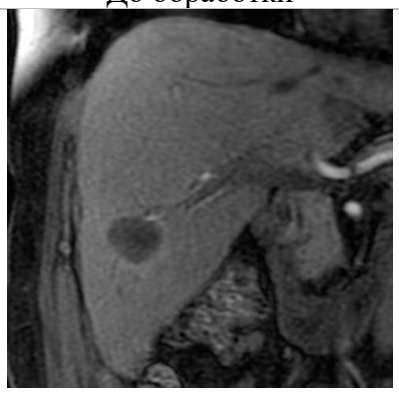
```
cv2.imshow('origin', my_photo)
cv2.waitKey()
cv2.destroyAllWindows()
```

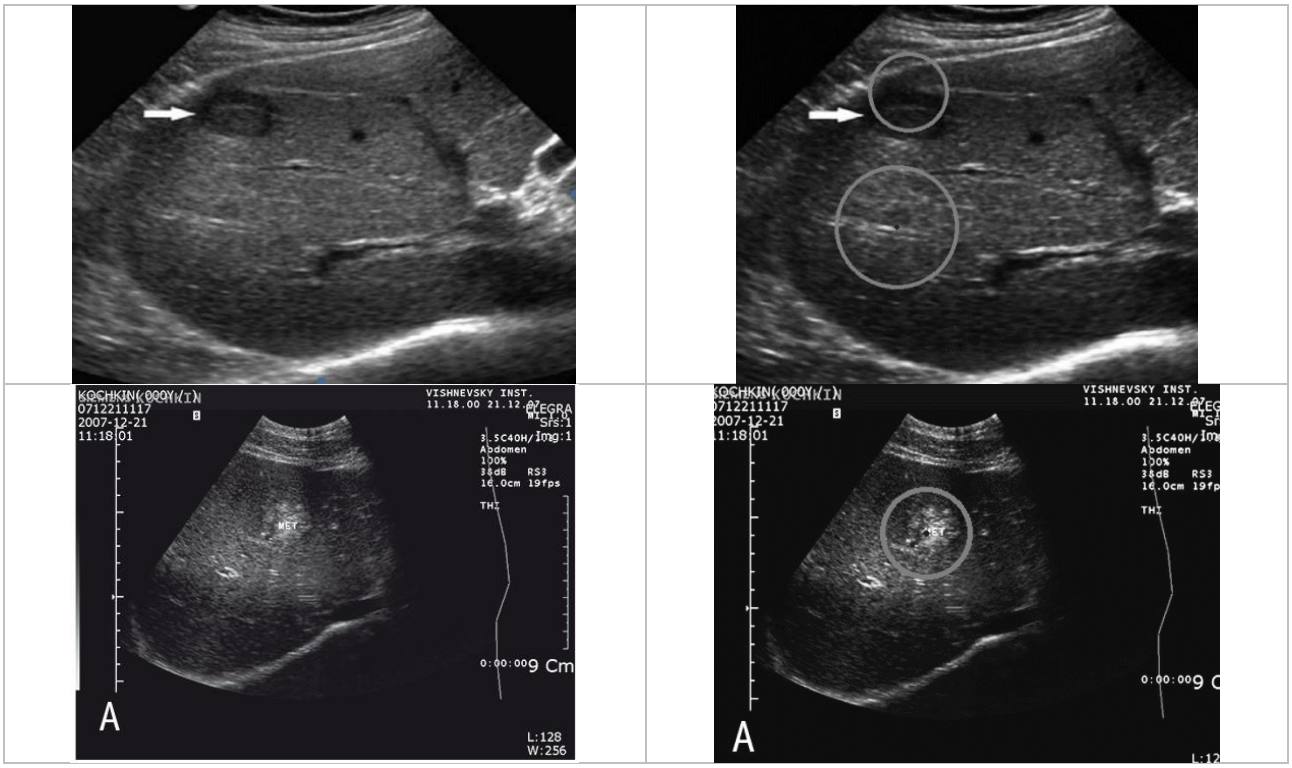
Рис. 12 – вывод итогового результата

Выполнение данной части кода предоставит серию изображений в отдельном окне. Нажимая клавишу «Esc», одно изображение будет сменяться следующим, по ходу выполнения программы.

Результат до и после обработки анонимных изображений посредством использованного кода представлен в таблице 1.

Таблица 1

УЗ-изображения до и после применения программного кода	
До обработки	После обработки
	



Полученный результат доказывает целесообразность использования представленного метода для обработки изображений УЗИ печени. Программа способна определить локализацию новообразования и обозначить подозрительные объекты, на которые специалисту стоит обратить внимание. Это упростит визуальное восприятие и увеличит работоспособность медицинского персонала, а также повысит возможность обследования большего числа пациентов.

Список литературы:

1. Горячкин Б.С., Китов М.А. КОМПЬЮТЕРНОЕ ЗРЕНИЕ // E-Scio. 2020. №9 (48).

2. Александровская Анна Андреевна, Маврин Евгений Михайлович Сравнение алгоритмов эквализации гистограмм полутоновых изображений // Вопросы науки и образования. 2019. №13 (60).

3. Шеладия Маноджкумар, Ачарья Шейли, Котари Ашиш, Ачарья Ганшьям ПРИМЕНЕНИЕ ТЕХНИКИ ЦИФРОВОЙ ОБРАБОТКИ ИЗОБРАЖЕНИЙ В АНАЛИЗЕ МИКРОСТРУКТУРЫ И ИССЛЕДОВАНИИ ОБРАБАТЫВАЕМОСТИ // Обработка металлов: технология, оборудование, инструменты. 2021. №4.

4. Толстунов Владимир Андреевич Сглаживающий фильтр с обобщенным гауссовским весом // Символ науки. 2018. №1-2.

5. Назарова Т.Ю., Лавров Д.Н. Компьютерное моделирование идентификации личности по радужной оболочке глаза на основе OpenCV // МСиМ. 2018. №3 (47).