Содержание

[Введение 5](#_Toc59095582)

[1. Теоретическая часть 6](#_Toc59095583)

[2. Программная реализация 13](#_Toc59095584)

[3. Тестирование 18](#_Toc59095585)

[Заключение 27](#_Toc59095586)

[Список использованных источников 28](#_Toc59095587)

## Введение

Что такое распознавание изображений и зачем оно нужно?

Актуальность этого процесса.

Цель распознавания изображений и применение в жизни.

Существующие библиотеки и уже написанный код

Цель работы – обзор методов tensorflow для распознавания объектов на изображении и классификации изображений

Задачи на научно–исследовательскую работу:

- разобрать методы для выделения объектов на изображении;

- разобрать методы для классификации изображений.

## Теоретическая часть

На изучение была предоставлена статья, написанная Simina Vasilache, Kevin Ward, Charles Cockrell, Jonathan Ha и Kayvan Najarian [1], в которой представлены алгоритмы морфологической предобработки изображений и изменение коэффициентов изображений, полученных после вейвлет-преобразований.

Вейвлет−преобразование - это математический инструмент для иерархической декомпозиции функций. С помощью вейвлетов функции представляются как композиция грубой низкочастотной аппроксимации и уточняющих компонент (деталей), представляющих отсутствующие в аппроксимации элементы графика функции. Вне зависимости от вида функции (изображение, кривая, поверхность) вейвлет представляет функцию как иерархию уровней отображения с различной точностью детализации. В процедурах предобработки изображений вейвлет-преобразование используется для уменьшения уровня шумов, анализа текстур, выделения контуров объектов и сжатия изображений.

Вейвлет Хаа́ра — один из первых и наиболее простых вейвлетов. Он основан на ортогональной системе функций, предложенной венгерским математиком Альфредом Хааром в 1909 году. Преобразование Хаара используется для сжатия входных сигналов, компрессии изображений, в основном цветных и черно-белых с плавными переходами. Идеален для картинок типа рентгеновских снимков. Данный вид архивации известен довольно давно и напрямую исходит из идеи использования когерентности областей.

Родительская (материнская) вейвлет-функция  с нулевым значением интеграла , определяющая детали сигнала, задается следующим образом:

 (1)

На этапе предварительной обработки проводится различие между брюшной областью и окружающими артефактами, такими как таблица КТ, кабели и нижние конечности пациента, которые видны на некоторых изображениях. Авторы статьи предлагают осуществлять сегментацию брюшной области от фоновых объектов следующим образом:

a) необработанное изображение бинаризуется (все пиксели с уровнем серого выше нуля становятся равными 1, пиксели с нулевым значением серого остаются в новом изображении равными 0;

b) к бинаризованному изображению применяются морфологические операции таким образом, чтобы различные объекты изображения были разделены;

c) чтобы выбрать объект с наибольшей площадью, необходимо произвести blob-анализ [2];

d) далее необходимо определить край объекта;

e) по этому краю нужно создать маску.

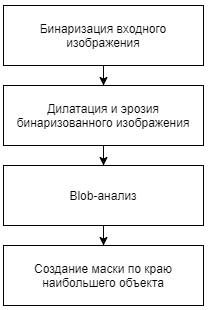


Рисунок 1 – Алгоритм предобработки

После применения получившейся маски устраняется фон, и в центре внимания метода становится сегментация кости. Изображение живота будет называться «исходным изображением». Шаги, предпринимаемые для получения первоначальной сегментации кости, следующие:

a) к предобработанному изображению применяется фильтр Гаусса;

b) на втором шаге к полученному изображению применяется вейвлет-анализ, затем, используя только матрицу аппроксимации, необходимо восстановить изображение;

c) вычисляется среднее значение и отклонение значений шкалы серого, отличных от нуля, то есть пикселей, не относящихся к фоновым;

d) расчет порогового значения как суммы средних и стандартного отклонения значений шкалы серого, рассчитанных на шаге c;

Пороговое значение *t1* представляет собой сумму модифицированных средних *m1* и стандартного отклонения *st1* значений серого пикселей, которые не являются фоном:

 (2)

где

 (3)

и

 (4)

 - это набор фоновых пикселей с нулевым значением уровня серого. *Card(S)* - это обозначение, используемое для мощности множества *S*.

e) создание начальной бинарной маски для кости, установление порогового значения для изображения в нижних частотах:

 (5)

f) определение ограничивающей рамки для области кости и ограничение последующей сегментации идентифицированной ограничивающей рамкой;

g) выравнивание гистограммы на изображении.

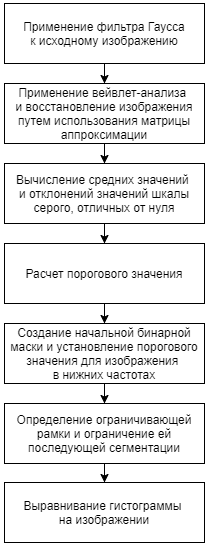


Рисунок 2 – Алгоритм сегментации

Следующий шаг - формирование контуров. Работа алгоритма, представленного в этом разделе, заключается в уточнении исходной маски кости в улучшенную маску промежуточной кости. Чтобы улучшить контуры изображения, применяется двумерный фильтр Гаусса. Ниже приводится краткое описание шагов на этом этапе:

a) определение оптимального размера окна фильтра на основе  - стандартного отклонения значений уровней серого пикселей изображения;

b) применение оптимального двухмерного фильтра Гаусса к копии исходного изображения;

Значения пикселей результирующего изображения *yg* определяются как:

 (6)

c) получение промежуточной маски кости путем двоичного умножения исходной маски кости и изображения, отфильтрованного по Гауссу:

 (7)



Рисунок 3 –Алгоритм формирование контуров

Выращивание контуров - заключительный этап сегментации. Чтобы автоматически получить маску кости, промежуточная маска умножается на копию исходного изображения после фильтрации SRAD. Выходное изображение получается из исходного изображения следующим образом:

 (8)

где - это оригинальное изображение,  - это выходное изображение,  - опора изображения,  - граница ,  - внешняя нормаль к , а *c(q)* - мгновенный коэффициент вариации.

Хорошие результаты сегментации дает выращивание контуров из костной маски по соседству. Техника выращивания описана ниже:

a) для каждого контура в маске кости идентифицируются соседи в окрестности *n × n* (в этом исследовании *n* = 3);

b) для каждого контура в маске кости значения градиента по восьми возможным направлениям вычисляются в окрестности *n × n*;

c) для каждого из соседей в окрестности *n × n* определяется их окрестность *m × m* (в данном исследовании *m* = 9);

d) вычисляется среднее значение уровня серого для соседей в окрестности *m × m*;

e) решение о добавлении соседа семени в маску кости принимается на основании двух условий:

- значение уровня серого соседа больше, чем среднее значение уровня серого соседа в окрестности *m × m*;

- значение градиента, соответствующее соседу, больше или равно -1.

Этот консервативный критерий предназначен для предотвращения роста контуров за пределы краев кости и сводит к минимуму риск слияния отдельных костей.

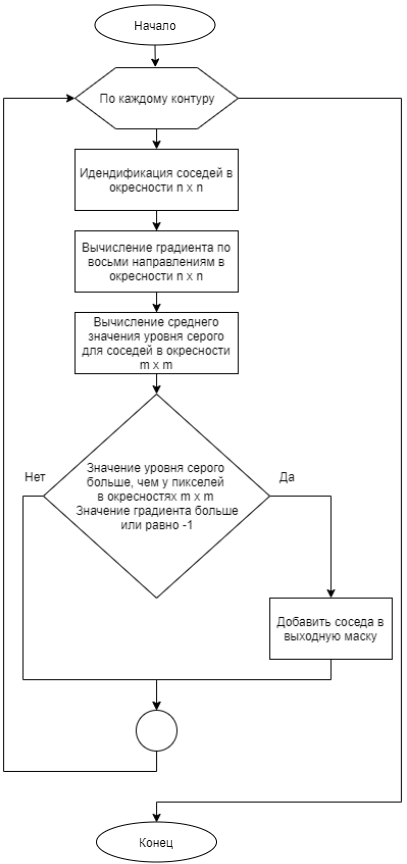


Рисунок 4 – Блок-схема «выращивания» контуров

## Программная реализация

Листинг программы:

import cv2

import numpy as np

import math

from pywt import dwt2, idwt2

import PIL

def Mask(image, mask):

mask[mask>0] = 1

return cv2.bitwise\_and(image, image, mask = mask)

def Morth(image):

dialated = cv2.dilate(image, cv2.getStructuringElement(cv2.MORPH\_DILATE,(7,7)), iterations = 1)

eroded = cv2.erode(dialated, cv2.getStructuringElement(cv2.MORPH\_ERODE,(7,7)), iterations = 2)

return eroded

def OnlyBiggestArea(image, mask):

cnts = cv2.findContours(mask, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

cnts = cnts[0] if len(cnts) == 2 else cnts[1]

max\_area = max([cv2.contourArea(c) for c in cnts])

for c in cnts:

area = cv2.contourArea(c)

if area != max\_area:

cv2.drawContours(mask, [c], -1, (0,0,0), -1)

return Mask(image, mask)

def contrast(image, coefficient):

avg = image.mean()

image = avg + coefficient \* (image - avg)

image[image>255] = 255

image[image<0] = 0

image = image.astype(np.uint8)

return image

def \_\_show(img):

cv2.imshow('show', cv2.resize(img, (500, 500)))

cv2.waitKey(0)

def cAcHcVcD\_change(cA, cH, cV, cD):

cA = np.zeros\_like(cA)

cH\_mean = cH.mean()

cV\_mean = cV.mean()

cH[cH < cH\_mean / 2 + cH\_mean] = 0

cV[cV < cV\_mean / 2 + cV\_mean] = 0

cH\_Gauss = cv2.GaussianBlur(cH, (7,7), 1.0)

cV\_Gauss = cv2.GaussianBlur(cV, (7,7), 1.0)

return cA, cH\_Gauss, cV\_Gauss, cD

def inversion(image):

image[image > 0] = 1

image[image == 0] = 255

image[image == 1] = 0

return image

def getContour(image\_input):

# ------------------------------ Pre-processing ------------------------------ #

image\_input\_contrast = contrast(image\_input, 2)

\_, tresholded = cv2.threshold(image\_input\_contrast, 0, 255, cv2.THRESH\_OTSU)

morth\_image = Morth(tresholded)

image\_only\_area = OnlyBiggestArea(image\_input\_contrast, mask=morth\_image)

# -------------------------- Initial mask formation -------------------------- #

img\_gauss = cv2.GaussianBlur(image\_only\_area, (9,9), 1.0)

cA, (cH, cV, cD) = dwt2(img\_gauss, 'haar')

cA\_2, (cH\_2, cV\_2, cD\_2) = dwt2(cA, 'haar')

cA\_2, cH\_2, cV\_2, cD\_2 = cAcHcVcD\_change(cA\_2, cH\_2, cV\_2, cD\_2 )

cA = idwt2((cA\_2, (cH\_2, cV\_2, cD\_2)), 'haar')

mask\_from\_dwt2 = idwt2((cA, (np.zeros\_like(cA), np.zeros\_like(cA), np.zeros\_like(cA))), 'haar')

mask\_from\_dwt2 = mask\_from\_dwt2.astype(np.uint8)

# \_\_show(mask\_from\_dwt2)

morth\_mask\_from\_dwt2 = cv2.dilate(mask\_from\_dwt2, cv2.getStructuringElement(cv2.MORPH\_DILATE,(11,11)), iterations = 2)

morth\_mask\_from\_dwt2\_inverted = inversion(morth\_mask\_from\_dwt2)

yellow\_image = np.zeros((image\_input.shape[0], image\_input.shape[1], 3), dtype='uint8')

yellow\_image[:, :, 0] = image\_input

yellow\_image[:, :, 1] = image\_input

yellow\_image[:, :, 2] = image\_input

for x in range(yellow\_image.shape[0]):

for y in range(yellow\_image.shape[1]):

if morth\_mask\_from\_dwt2\_inverted[x,y] == 255 and morth\_image[x,y] != 0:

yellow\_image[x,y,1] = 255

yellow\_image[x,y,2] = 255

return yellow\_image

Более наглядно алгоритм работы программы представлен на блок-схеме на рисунке 5.



Рисунок 5 – Блок-схема работы программы

Методы, используемые в данной программе, более подробно представлены в блок-схемах на рисунках 6, 7, 8, 9 и 10.

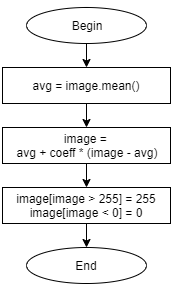


Рисунок 6 – Блок-схема алгоритма метода contrast

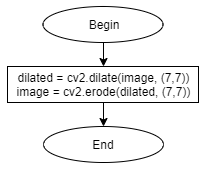


Рисунок 7 – Блок-схема алгоритма метода Morth

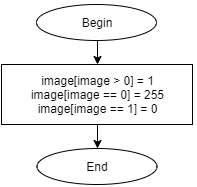


Рисунок 8 – Блок-схема алгоритма метода inversion

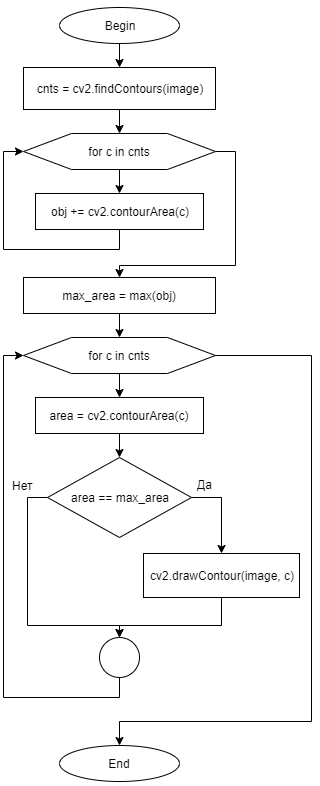


Рисунок 9 – Блок-схема метода OnlyBiggestArea

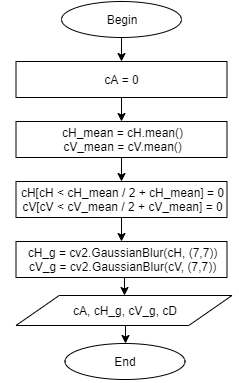


Рисунок 10 – Блок-схема метода cAcHcVcD\_change

## 3. Тестирование

Таблица – Результаты обработки изображений

|  |  |  |
| --- | --- | --- |
| Название | Исходное изображение | Обработанное изображение |
| mdb028 |  |  |
| mdb028\_1 |  |  |
| mdb032 |  | C:\Users\minir\AppData\Local\Microsoft\Windows\INetCache\Content.Word\mdb032.bmp_filled.bmp |
| mdb032\_1 |  |  |
| mdb058 |  |  |

Продолжение таблицы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| mdb058\_1 | |  | |  | |
| mdb075 | |  | |  | |
| mdb075\_1 | |  | |  | |
| mdb092 | |  | |  | |
| mdb092\_1 | |  | |  | |

Продолжение таблицы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| mdb095 | |  | |  | |
| mdb095\_1 | |  | |  | |
| mdb102 | |  | |  | |
| mdb102\_1 | |  | |  | |
| mdb104 | |  | |  | |

Продолжение таблицы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| mdb104\_1 | |  | |  | |
| mdb105 | |  | |  | |
| mdb105\_1 | |  | |  | |
| mdb110 | |  | |  | |
| mdb110\_1 | |  | |  | |

Продолжение таблицы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| mdb111 | |  | |  | |
| mdb111\_1 | |  | |  | |
| mdb115 | |  | |  | |
| mdb115\_1 | |  | |  | |
| mdb117 | |  | |  | |

Продолжение таблицы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| mdb177\_1 | |  | |  | |
| mdb120 | |  | |  | |
| mdb120\_1 | |  | |  | |
| mdb124 | |  | |  | |
| mdb124\_1 | |  | |  | |

Продолжение таблицы

|  |  |  |  |
| --- | --- | --- | --- |
| mdb134 |  |  | |
| mdb134\_1 |  |  | |
| mdb141 |  |  | |
| mdb141\_1 |  | |  |
| mdb144 |  | |  |

Продолжение таблицы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| mdb144\_1 | |  | |  | |
| mdb145 | |  | |  | |
| mdb145\_1 | |  | |  | |
| mdb155 | |  | |  | |
| mdb155\_1 | |  | |  | |

Окончание таблицы

|  |  |  |
| --- | --- | --- |
| mdb158 |  |  |
| mdb158\_1 |  |  |

Методом, представленным в статье, можно сегментировать опухоль на полноразмерных маммологических снимках. Верно опухоль была сегментирована на 21 полноразмерных снимках из 21. Метод, представленный в статье, не обладает достаточными алгоритмами предобработки изображения для сегментации опухолей на макроснимках, так как из 21 представленных макроснимков верно опухоль была сегментированга лишь на 5 снимках.

## Заключение

Используя вейвлет-преобразование, можно сгладить или выделить некоторые детали изображения, которые впоследствии можно будет сегментировать от исходного изображения.

В ходе выполнения научно-исследовательской работы были достигнуты следующие цели:

- алгоритм, описанный в предоставленной статье, изучен;

- представленный алгоритм сегментации реализован на маммограммах;

- тестирование реализованного алгоритма проведено на маммографических снимках.

Тестирование показало, что данный алгоритм на различных изображениях работает по-разному.

Результаты тестирования, следующие:

- опухоли на макроснимках выделяются плохо из-за отсутствия достаточной предобработки исходных изображений, алгоритмы которой не предоставлены в статье,

- работа представленного в статье алгоритма качественно происходит только на полноразмерных маммографических снимках.

Таким образом, можно сделать вывод о том, что цель работы выполнена, все поставленные задачи реализованы в полном виде.

## Список использованных источников

1. Simina Vasilache, Kevin Ward, Charles Cockrell, Jonathan Ha, Kayvan Najarian. Unified wavelet and gaussian filtering for segmentation of CT images; application in segmentation of bone in pelvic CT images // BMC Medical Informatics and Decision Making. 2009. 8 с. DOI:10.1186/1472-6947-9-S1-S8.

2. Feature Detection and Description [электронный ресурс]. Режим доступа: https://opencv-python-tutroals.readthedocs.io (дата обращения: 16.12.20).

3. Сегментация изображения с OpenCV и Python [электронный ресурс]. Режим доступа: https://pythonru.com/biblioteki/segmentacija-izobrazhenija-s-opencv-i-python.html (дата обращения: 16.12.20).