**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ**

**Московский государственный технический университет**

**им. Н.Э. Баумана**

**(МГТУ им. Н.Э. Баумана)**

**Кафедра «Информационная системы и технологии» (ИУ3)**

Лабораторная работа № 1

По дисциплине: «Цифровая обработка изображений»

# Тема: «Распознавание объекта по форме»

Выполнили: Кисарева Н.Ю., Быковский М.К.

студенты группы ИУ3-22М

Проверил: Алфимцев А.Н.

Профессор каф. ИУ3

Москва, 2018 г.

## Задание

1. Получить обучающую и тестовую выборки изображений двух фруктов у преподавателя.
2. В среде Spyder (сборка Anaconda) на языке Python (3.5 и старше) создать проект и подключить библиотеку *scikit-image.*
3. Разработать программу по загрузке цифрового изображения.
4. Подключить функцию обращения к пикселям цифрового изображения.
5. Запрограммировать формулу перевода цифрового изображения в полутоновой формат.
6. Провести информационный поиск в Интернете по способам применения оператора выделения границ в соответствии с вариантом (В1: Кэнни, В2: Прюитт, В3: Робертс, В4: Собель, В5: Аверсон, В6: Ротуэлл). Разобраться, включить описание в отчет (теор. часть).
7. Запрограммировать оператор выделения границ в соответствии с вариантом. Вариант - Кэнни
8. Запрограммировать работу программы распознавания фрукта в зависимости от формы объекта из обучающей выборки. Используемые фрукты – авокадо и яблоко
9. Организовать и провести эксперимент на тестовой выборке. Вычислить ошибку первого и второго рода.
10. Дать комментарий каждой строчке кода

## Теоретическая часть

**Оператор Кэнни** (**детектор границ Кэнни**, **алгоритм Кэнни**) в дисциплине [компьютерного зрения](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%BE%D0%B5_%D0%B7%D1%80%D0%B5%D0%BD%D0%B8%D0%B5) — оператор [обнаружения границ](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B3%D1%80%D0%B0%D0%BD%D0%B8%D1%86) изображения. Был разработан в [1986 году](https://ru.wikipedia.org/wiki/1986_%D0%B3%D0%BE%D0%B4) [Джоном Кэнни](https://ru.wikipedia.org/w/index.php?title=%D0%9A%D1%8D%D0%BD%D0%BD%D0%B8,_%D0%94%D0%B6%D0%BE%D0%BD&action=edit&redlink=1) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *[John F. Canny](https://en.wikipedia.org/wiki/John_F._Canny" \o "en:John F. Canny)*) и использует многоступенчатый [алгоритм](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC" \o "Алгоритм)для обнаружения широкого спектра границ в изображениях.

Кэнни изучил математическую проблему получения [фильтра](https://ru.wikipedia.org/wiki/%D0%A4%D0%B8%D0%BB%D1%8C%D1%82%D1%80_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), оптимального по критериям выделения, локализации и минимизации нескольких откликов одного края. Он показал, что искомый фильтр является суммой четырёх экспонент. Он также показал, что этот фильтр может быть хорошо приближен первой производной Гауссианы. Кэнни ввёл понятие *подавления немаксимумов* ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Non-Maximum Suppression*), которое означает, что [пикселями](https://ru.wikipedia.org/wiki/%D0%9F%D0%B8%D0%BA%D1%81%D0%B5%D0%BB%D1%8C) границ объявляются пиксели, в которых достигается локальный максимум градиента в направлении вектора градиента.

Хотя его работа была проведена на заре компьютерного зрения, детектор границ Кэнни до сих пор является одним из лучших детекторов. Кроме особенных частных случаев трудно найти детектор, который бы работал существенно лучше, чем детектор Кэнни.

Целью Кэнни было разработать оптимальный алгоритм обнаружения границ, удовлетворяющий трём критериям:

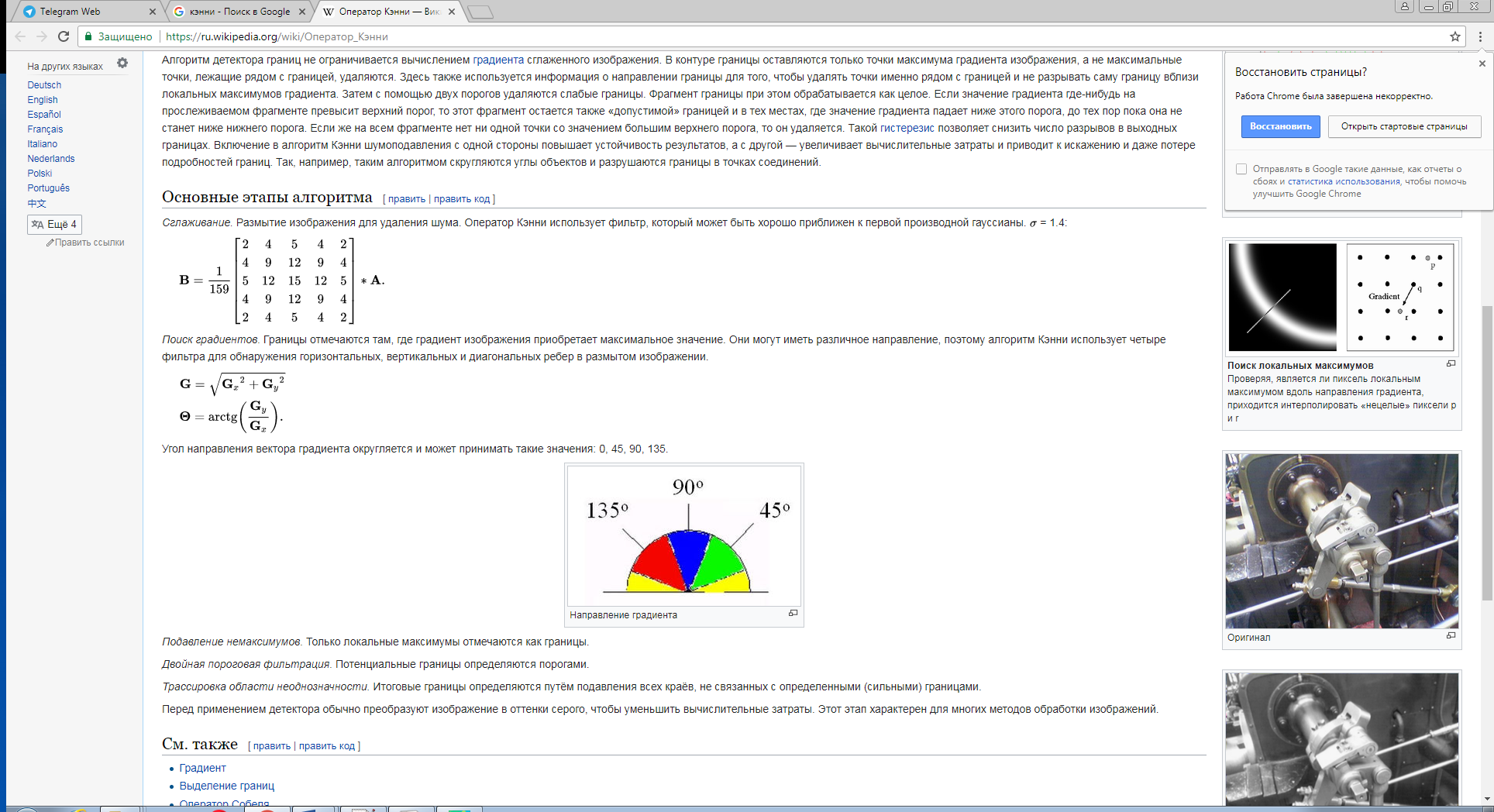
* хорошее обнаружение (Кэнни трактовал это свойство как повышение отношения [сигнал](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%B3%D0%BD%D0%B0%D0%BB)/[шум](https://ru.wikipedia.org/wiki/%D0%A6%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%BE%D0%B9_%D1%88%D1%83%D0%BC_%D0%B8%D0%B7%D0%BE%D0%B1%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F));
* хорошая локализация (правильное определение положения границы);
* единственный отклик на одну границу.

Из этих критериев затем строилась целевая функция стоимости ошибок, минимизацией которой находится «оптимальный» линейный оператор для свёртки с изображением.

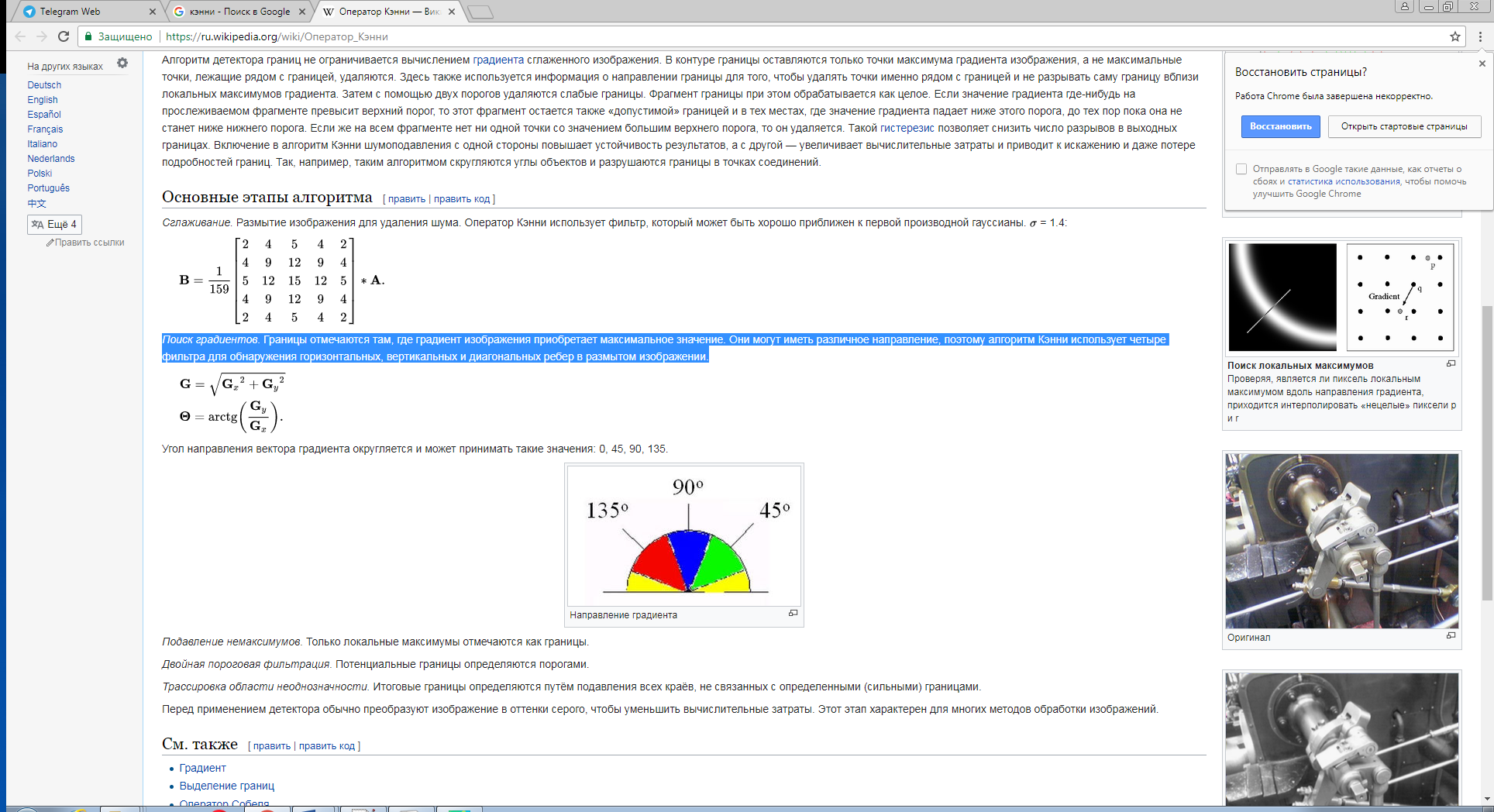
Алгоритм детектора границ не ограничивается вычислением [градиента](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D0%B4%D0%B8%D0%B5%D0%BD%D1%82) сглаженного изображения. В контуре границы оставляются только точки максимума градиента изображения, а не максимальные точки, лежащие рядом с границей, удаляются. Здесь также используется информация о направлении границы для того, чтобы удалять точки именно рядом с границей и не разрывать саму границу вблизи локальных максимумов градиента. Затем с помощью двух порогов удаляются слабые границы. Фрагмент границы при этом обрабатывается как целое. Если значение градиента где-нибудь на прослеживаемом фрагменте превысит верхний порог, то этот фрагмент остается также «допустимой» границей и в тех местах, где значение градиента падает ниже этого порога, до тех пор пока она не станет ниже нижнего порога. Если же на всем фрагменте нет ни одной точки со значением большим верхнего порога, то он удаляется. Такой [гистерезис](https://ru.wikipedia.org/wiki/%D0%93%D0%B8%D1%81%D1%82%D0%B5%D1%80%D0%B5%D0%B7%D0%B8%D1%81) позволяет снизить число разрывов в выходных границах. Включение в алгоритм Кэнни шумоподавления с одной стороны повышает устойчивость результатов, а с другой — увеличивает вычислительные затраты и приводит к искажению и даже потере подробностей границ. Так, например, таким алгоритмом скругляются углы объектов и разрушаются границы в точках соединений.

## **Основные этапы алгоритма**

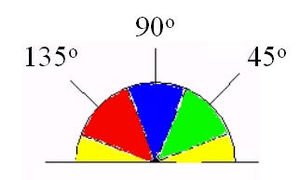
*Сглаживание.* Размытие изображения для удаления шума. Оператор Кэнни использует фильтр, который может быть хорошо приближен к первой производной гауссианы. {\displaystyle \sigma }σ = 1.4:



*Поиск градиентов.* Границы отмечаются там, где градиент изображения приобретает максимальное значение. Они могут иметь различное направление, поэтому алгоритм Кэнни использует четыре фильтра для обнаружения горизонтальных, вертикальных и диагональных ребер в размытом изображении.



Угол направления вектора градиента округляется и может принимать такие значения: 0, 45, 90, 135.

[](https://commons.wikimedia.org/wiki/File:Semicirc.jpg?uselang=ru)

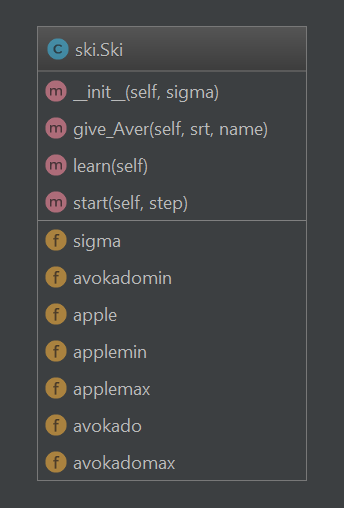
*Подавление немаксимумов.* Только локальные максимумы отмечаются как границы.

*Двойная пороговая фильтрация.* Потенциальные границы определяются порогами.

*Трассировка области неоднозначности.* Итоговые границы определяются путём подавления всех краёв, не связанных с определенными (сильными) границами.

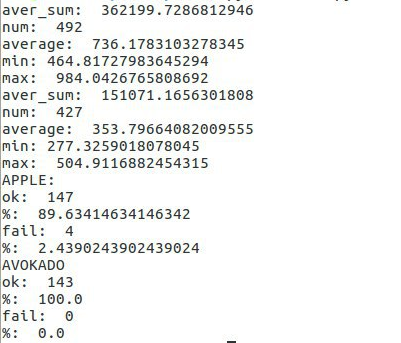
Перед применением детектора обычно преобразуют изображение в оттенки серого, чтобы уменьшить вычислительные затраты. Этот этап характерен для многих методов обработки изображений.

## Диаграмма структуры программы



## Интерфейс и основные шаги работы программы

В результате работы программы мы получаем следующий результат:



На первом шаге мы инициализируем сеть. Далее для каждой из двух выборок считаем средний периметр, максимальное значение и минимальное. На основе рассчитанных данных проводим распознавание тестовой выборки. Результаты распознавания представлены выше. На основе минимальных и максимальных значений возможно регулирование и подстройка промежутка, при попадании в который мы делаем вывод о принадлежности изображения.

## Листинг программы

ski.py:

import skimage *#подключаем библиотеки*import os  
from skimage import data, io, filters, color  
from skimage import feature  
from skimage import measure  
from skimage.color import rgb2gray  
  
class Ski: *#объявляем класс* def \_\_init\_\_(self, sigma): *#конструктор с параметром* self.sigma = sigma *#сохраняем параметр в приватную переменную* def give\_Aver (self, srt, name): *#функция подсчета периметра изображения* filename = os.path.join(skimage.data\_dir, srt + name) *#подключаем путь к файлу* image = io.imread(filename) *#читаем изображение* image\_g = rgb2gray(image) *#переводим в серые тона* edges = feature.canny(image\_g, sigma=self.sigma) *#применяем канни* aver = measure.perimeter(edges) *#считаем периметр* return aver *#возвращаем результат* def learn (self): *#функция обучения* files = os.listdir('apple/') *#получаем список файлов в заданном пути* aver\_sum = 0 *#обнуляем параметры* num = 0  
 minn = 1000  
 maxx = 100  
 for name in files: *#пробегаемся по массиву файлов* aver = self.give\_Aver(srt = 'apple/', name = name) *#считаем среднее для текущего изображения массива* aver\_sum += aver *#прибавляем результат* num += 1 *#увеличиваем счетчик* if(minn > aver): *#считаем минимум, если текущее меньше сохраненного* minn = aver *#сохраняем* if(maxx < aver): *#считаем максимум, если текущее больше сохраненного* maxx = aver *#сохраняем* print("aver\_sum: ", aver\_sum) *#печатаем результат* print("num: ", num)  
 print("average: ", aver\_sum/num)  
 print("min:", minn)  
 print("max: ", maxx)  
 self.apple = aver\_sum/num *#сохраняем результат в приватную переменную* self.applemin = minn  
 self.applemax = maxx  
   
 files = os.listdir('avokado/') *#повтор предыдущего шага для нового пути* aver\_sum = 0  
 num = 0  
 minn = 1000  
 maxx = 100  
 for name in files:  
 aver = self.give\_Aver(srt = 'avokado/', name = name)  
 aver\_sum += aver  
 num += 1  
 if(minn > aver):  
 minn = aver  
 if(maxx < aver):  
 maxx = aver  
  
 print("aver\_sum: ", aver\_sum)  
 print("num: ", num)  
 print("average: ", aver\_sum/num)  
 print("min:", minn)  
 print("max: ", maxx)  
 self.avokado = aver\_sum/num  
 self.avokadomin = minn  
 self.avokadomax = maxx  
  
 def start(self, step): *#запуск распознавания* files = os.listdir('testing/apple/') *#открываем файлы по пути, полностью по аналогии с функцией обучения* ok = 0  
 fail = 0  
 num = 0  
  
 for name in files:  
 aver = self.give\_Aver(srt = 'testing/apple/', name = name) *#получаем периметр* num += 1  
 if(self.apple - step < aver and self.apple + step > aver): *#если он находится в заданном промежутке от рассчитанного при обучении среднего, то распознание верно* ok += 1  
 elif(self.avokado - step < aver and self.avokado + step > aver): *#если находится в другом промежутке, значит не верно.* fail += 1  
 *#стоит добавить, что в данном случае распознавание организовано с точки зрения проверки правильности разпознавания. В боевом режиме просто используем два промежутка.  
 #на качество распознавания это никак не влияет!* print("APPLE:") *#выводим результат* print("ok: ", ok)  
 print("%: ", ok/num\*100)  
 print("fail: ", fail)  
 print("%: ", fail/num\*100)  
  
 files = os.listdir('testing/avokado/') *#полный повтор для другого пункта* ok = 0  
 fail = 0  
 num = 0  
  
 for name in files:  
 aver = self.give\_Aver(srt = 'testing/avokado/', name = name)  
 num += 1  
 if(self.avokado - step < aver and self.avokado + step > aver):  
 ok += 1  
 elif(self.apple - step < aver and self.apple + step > aver):  
 fail += 1  
  
 print("AVOKADO")  
 print("ok: ", ok)  
 print("%: ", ok/num\*100)  
 print("fail: ", fail)  
 print("%: ", fail/num\*100)

lab.py:

import ski *#подключаем написанную библиотеку*neuro = ski.Ski(sigma=2) *#создаем экземпляр класса с параметром*neuro.learn() *#запускаем обучение сети*neuro.start(150) *#запускаем сеть с параметром интервал поиска вхождений*

## Список использованной литературы

1. <https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80_%D0%9A%D1%8D%D0%BD%D0%BD%D0%B8> - Электронный ресурс – дата обращения 15.02.2018 г.
2. https://habrahabr.ru/post/114589/ – Электронный ресурс – дата обращения 15.02.2018 г.