ГУАП

ГУАП

guap.ru

Сегментация изображений. Метод К-средних.

Студент группы 5120М П.Н. Сергеева
Преподаватель Е.Д. Пойманова

Сегментация изображения

Сегментация изображения - это процесс разделения цифрового изображения на несколько отдельных областей, содержащих каждый пиксель (наборы пикселей) с аналогичными атрибутами.
Цель сегментации изображения состоит в том, чтобы изменить представление изображения во что-то более значимое и более простое для анализа.
Сегментация изображения обычно используется для поиска объектов и границы(линии, кривые и т. д.) в изображениях. Точнее говоря, сегментация изображения - это процесс назначения метки каждому пикселю в изображении, чтобы пиксели с одинаковой меткой имели определенные характеристики. Многие виды исследований были проведены в области сегментации изображений с использованием кластеризации. Существуют разные методы, и один из самых популярных методов Алгоритм кластеризации K-средних.

Сегментация изображения

Почему сегментация изображения имеет значение?

Если мы возьмем пример автономных транспортных средств, им нужны сенсорные устройства ввода, такие как камеры, радары и лазеры, чтобы позволить автомобилю воспринимать окружающий мир, создавая цифровую карту. Автономное вождение невозможно даже без обнаружения объекта, что само по себе включает классификацию / сегментацию изображения.
К другим примерам относится индустрия здравоохранения, где, если мы говорим о раке, даже в нынешний век технологических достижений рак может быть смертельным, если мы не идентифицируем его на ранней стадии. Обнаружение раковых клеток как можно быстрее может спасти миллионы жизней. Форма раковых клеток играет жизненно важную роль в определении тяжести рака, которая может быть идентифицирована с использованием алгоритмов классификации изображений.

Алгоритм k-средних

K-средних – простой и эффективный алгоритм, который легко реализовать программным методом. Алгоритм является неконтролируемым алгоритмом и используется для отделения интересующей области от фона. Он группирует или разделяет данные на K-кластеры или части на основе K-центроидов.

Алгоритм используется, когда у вас есть немеченые данные (то есть данные без определенных категорий или групп). Цель состоит в том, чтобы найти определенные группы на основе некоторого сходства данных с количеством групп, представленных K.

Алгоритм k-средних

1. Нужно наперед знать в сколько кластеров нужно распределить данные. Это является существенным минусом данного метода.
2. Нужно выбрать начальные центры наших кластеров. Именно «расстояние» от центра до пикселя определяет, какому кластеру будет принадлежать каждый пиксель.
3. Посчитаем расстояние от каждого центра до каждого пикселя. Это расстояние считается как евклидово расстояние между точками в пространстве, а в нашем случае — как расстояние между тремя составляющими цвета:

$$\sqrt{(R_2 - R_1)^2 + (G_2 - G_1)^2 + (B_2 - B_1)^2}.$$

Алгоритм k-средних

Считаем расстояние от первого пикселя до каждого центра и определяем наименьшее расстояние между этим пикселем и центрами. Для центра расстояние, до которого является наименьшим, пересчитываем координаты, как среднее арифметическое между каждой составляющей пикселя – короля и пикселя — подданного. Наш центр смещается в пространстве соответственно подсчетам.

4. После пересчета всех центров, мы распределяем пиксели по кластерам, сравнивая расстояние от каждого пикселя до центров. Пиксель помещается в кластер, к центру которого он расположен ближе, чем к остальным центрам.

5. Все начинается сначала, до тех пор, пока пиксели остаются в одних и тех же кластерах. Иногда при большом количестве данных центры будут перемещаться в малом радиусе, и пиксели по краям кластеров будут прыгать то в один, то в другой кластер. Для этого нужно определить максимальное число итераций

Алгоритм k-средних

Если говорить о K-средних, то правильный выбор K часто неоднозначен с интерпретациями, зависящими от формы и масштаба распределения точек в наборе данных и желаемого разрешения кластеризации пользователя. Кроме того, увеличение K без штрафа всегда будет уменьшать количество ошибок в результирующей кластеризации до крайнего случая нулевой ошибки. Тогда интуитивно оптимальный выбор K обеспечит баланс между максимальным сжатием данных с использованием одного кластера и максимальной точностью, назначая каждую точку данных своему кластеру,

Если подходящее значение K не очевидно из предшествующего знания свойств набора данных, его нужно каким-то образом выбрать. Есть несколько категорий методов для принятия этого решения и Метод локтя один из таких методов.

Реализация

Следующим шагом является загрузка изображения в цветовом пространстве RGB. Затем нам нужно преобразовать наше изображение из RGB Colors Space в HSV, чтобы работать дальше. Далее преобразуем изображение MxNx3 в матрицу Kx3, где K = MxN, и каждая строка теперь является вектором в трехмерном пространстве RGB.
И конвертируем значения unit8 в float.

```
# Загрузка изображения в цветовом пространстве RGB
original_image = cv2.imread("C:/a/test2.jpeg")

# Преобразование цветового пространства RGB в HSV
img=cv2.cvtColor(original_image,cv2.COLOR_BGR2RGB)
vectorized = img.reshape((-1,3))

# Преобразовать np.float32
vectorized = np.float32(vectorized)
```

Реализация алгоритма

Сгруппируем с k = 3, потому что если посмотреть на изображение, оно имеет 3 цвета, зеленую траву, белые лепестки и желтую сердцевину. Определим критерии и количество кластеров, и запустим алгоритм К-средних. Затем необходимо конвертировать обратно в uint8 и получить доступ к меткам, чтобы восстановить изображение и визуализировать его.

```
# Здесь мы применяем кластеризацию k-средних, чтобы пиксели вокруг цвета были согласованы и давали одинаковые значения RGB/HSV.
# определить критерии, количество кластеров и применить kmeans
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)

K = 3
attempts=10
ret,label,center=cv2.kmeans(vectorized,K,None,criteria,attempts,cv2.KMEANS_PP_CENTERS)

# Теперь конвертируем обратно в uint8
# теперь нам нужно получить доступ к меткам, чтобы восстановить кластеризованное изображение
center = np.uint8(center)
res = center[label.flatten()]
res2 = res.reshape((img.shape))
```

Реализация алгоритма

Как видно, алгоритм разделил наше исходное изображение на три доминирующих цвета при K = 3, а при K = 5 получили более точное изображение. С увеличением значения K, изображение становится более четким, поскольку алгоритм K-средних может классифицировать больше классов / кластеров цветов.

Сегментация изображения при K = 3

Сегментация изображения при K = 5

Реализация алгоритма

Обнаружение Canny Edge - это метод обработки изображения, используемый для обнаружения краев на изображении при подавлении шума.
Алгоритм обнаружения Canny Edge состоит из 5 этапов:
1.Подавление шума
2. Расчет градиента
3. Не максимальное подавление
4. Двойной порог
5. Отслеживание краев по гистерезису.

Функция находит ребра во входном изображении (8-битное входное изображение) и помечает их на ребрах выходной карты с помощью алгоритма Канни. Наименьшее значение между порогом 1 и порогом 2 используется для связывания ребер. Наибольшее значение используется для поиска начальных сегментов сильных ребер.

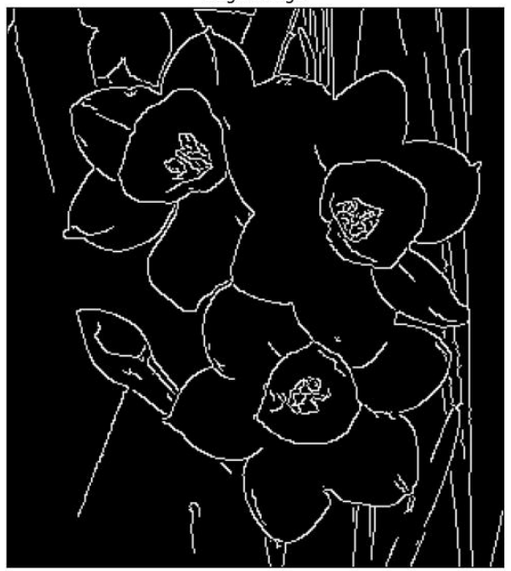
Реализация алгоритма

```
#обнаружение краев
edges = cv2.Canny(img,100,200)
plt.figure(figsize=(figure_size,figure_size))
plt.subplot(1,2,1),plt.imshow(img)
plt.title('Original Image'), plt.xticks([]), plt.yticks([])
plt.subplot(1,2,2),plt.imshow(edges,cmap = 'gray')
plt.title('Edge Image'), plt.xticks([]), plt.yticks([])

plt.show()
```

Original Image

Edge Image



Вывод

Сегментация является одной из основных задач обработки и анализа изображений. Цель сегментации заключается в упрощении и/или изменении представления изображения, чтобы его было проще и легче анализировать. И она будет полезна во многих областях(медицина, автономное вождение, обработка изображений со спутников).

ГУАП

guap.ru

Спасибо за внимание!

Сегментация изображения

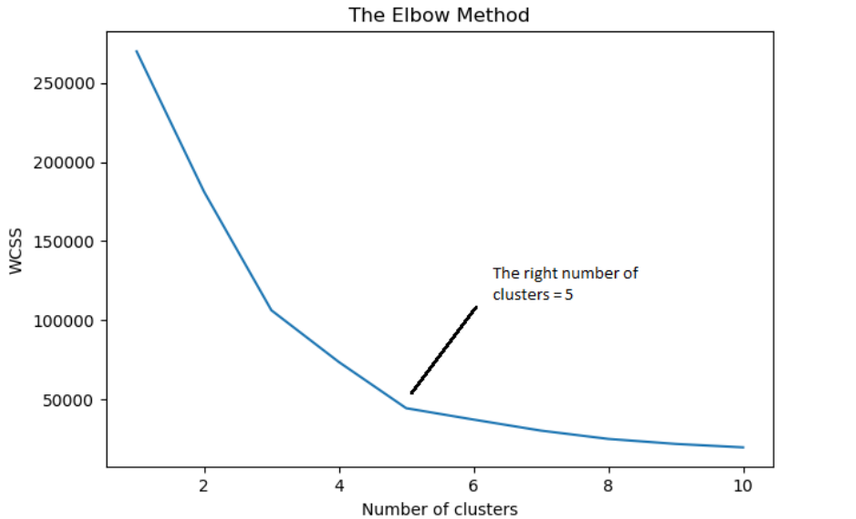
Сегментация изображения включает в себя преобразование изображения в набор областей пикселей, которые представлены маской или помеченным изображением. Разделив изображение на сегменты, вы можете обрабатывать только важные сегменты изображения вместо обработки всего изображения.

Обычный метод заключается в поиске резких разрывов в значениях пикселей, которые обычно указывают края, которые определяют область.

Другим распространенным подходом является обнаружение сходства в областях изображения. Некоторые методы, которые следуют этому подходу, включают в себя рост регионов, кластеризацию и пороговое определение..

Метод локтя

Основная идея методов разделения, таких как кластеризация K-Means, состоит в том, чтобы определить кластеры таким образом, чтобы общая вариация внутри кластера или, другими словами, общая сумма квадрата внутри кластера (WCSS) была минимизирована. Общий WCSS измеряет компактность кластеризации, и мы хотим, чтобы она была как можно меньше.



Метод локтя

Метод Elbow рассматривает общий WCSS как функцию от числа кластеров: нужно выбрать количество кластеров, чтобы добавление другого кластера не улучшило бы значительно WCSS в целом.

Шаги по выбору оптимального количества кластеров K: (метод Elbow)
1. Вычислите кластеризацию K-средних для различных значений K, варьируя K от 1 до 10 кластеров.
2. Для каждого K вычислите общую сумму квадрата внутри кластера (WCSS).
3. Постройте кривую WCSS против количества кластеров K.
4. Расположение изгиба (колена) на участке обычно рассматривается как показатель соответствующего количества кластеров.

Метод локтя

Несмотря на все преимущества, K-Means имеет, но иногда терпит неудачу из-за случайного выбора центроидов, который называется Случайная ловушка инициализации.
Чтобы решить эту проблему, у нас есть процедура инициализации для K-Means.
В K-Means ++ мы выбираем точку случайным образом, и это ваш первый центроид, затем мы выбираем следующую точку на основе вероятности, которая зависит от расстояния до первой точки, чем дальше друг от друга точка, тем более вероятна она.
Затем у нас есть два центроида, повторите процесс, вероятность каждой точки основана на расстоянии до ближайшего центроида к этой точке. В настоящее время,это приводит к накладным расходам при инициализации алгоритма, но снижает вероятность неправильной инициализации, приводящей к плохому результату кластеризации.