Data Science.
Lectures. Week 3-4.
Computer Vision. Компьютерное зрение.

конспект составил Кильдияров Тимур.

3 ноября 2022 г.

Lecture Topics

1 Что такое изображение

В общем случае под изображением подразумевают некоторую функцию от координат.

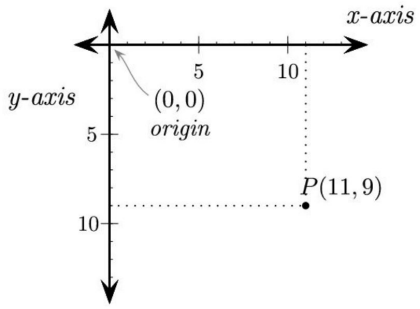
В рамках данной системы мы можем задать вопрос "какой цвет у картинки в данной точке?"

Изображение может храниться в нескольких форматах:

Растровый

Двумерный массив, для каждого пикселя в координатах x,y известен цвет. За нуль взят самый верхний левый угол

Координатная система



Координатная система от Chip

растровые изображения зачастую имеют вспомогательную информацию: мето съемки, время съемки, оборудовании для съемки. Эти вспомогательные данные могут помочь при обработке фотографии. Бывали случаи, когда с помощью данных о времени съемки, месте съемки и оборудовании для съемки получалось добиться результатов, которых бы не добились, имея только данные о двумерном массиве с цветами

Растровый формат используется для фотомонтажа. Все фотографии снимаются в растровом формате

Векторный

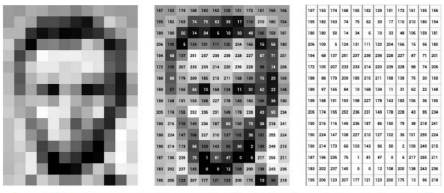
мы создаем картинку через некоторое правило. Как правило используем кривые Безье.

Используется для создания дизайна, к примеру сайта

в дальнейшем будем говорить про картинку в растровом формате

Пример растрового изображения:

Растровые изображения



Авраам Линькольн из мастер-класса от Stanford по фильтрации изображений

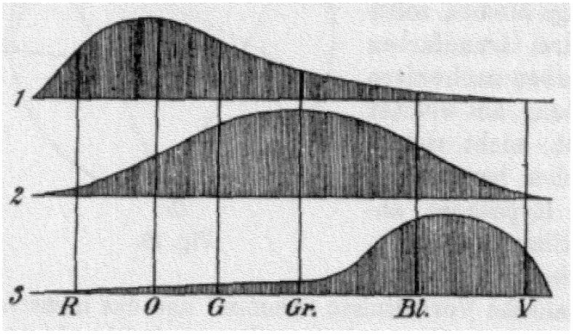
Пиксели с значением 255 - белые, а с 0 - чёрные. всего 256 значений, ибо так легче и проще

хранить картинки. Так удобно хранить картинки не только чёрно-белые, но и RGB.

Цветовые пространства

RGB

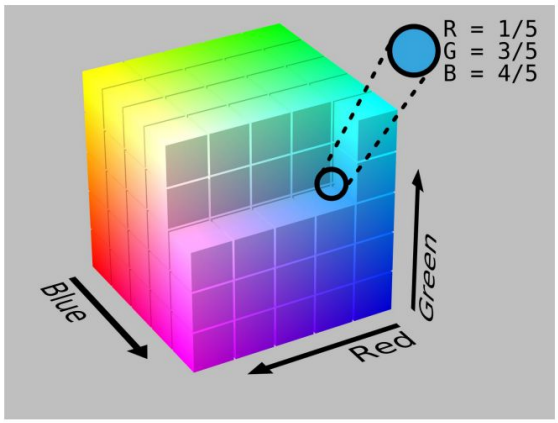
Задаём картинку через смесь красного, зеленого, синего цветов (все экраны мониторов делают с помощью RGB частиц).



Три разновидности рецепторов от Hermann von Helmhotz

Картинка выше показывает восприимчивость человеческого глаза к различным цветам. Видно, что наиболее восприимчив человеческий глаз к длинам волн красного, зеленого и синего цветов. Именно поэтому и возникло RGB.

У некоторых людей есть мутация, из-за чего они восприимчивы не только к красному, зеленому, синему, но и к ещё какому-то цвету или каким-то цветам.Для них современные мониторы отражают цвета не так хорошо, как они видят их в "природе"



RGB куб от SharkD, CC BY-SA 3.0

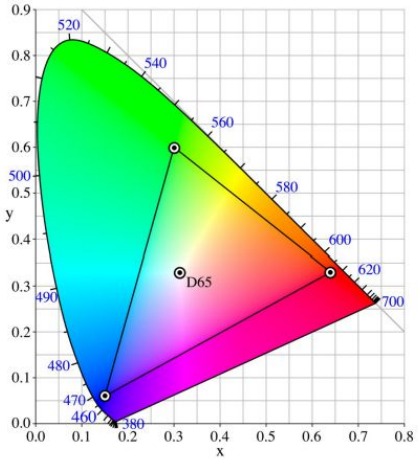
Кубик выше показывает как можно задать различные цвета при помощи добавления красного/синего/зелёного.

Первая картинка на странице 4 показывает, что RGB охватывает не все цвета, которые человек видит в "природе". Пустые квадратики - цвета, которые RGB пока не показывает.

Grayscale

Градации серого. Конвертация RGB в Grayscale иногда лучше для обучения различных моделей. Зачем моделе для распознования рукописных цифр RGB? Он не нужен, используем Grayscale! Grayscale позволяет хранить изображения в 3 раза меньше по весу (биты). RGB – трёхмерный массив от 0 до 255, а Grayscale – одномерный массив от 0 до 255.

Формула для линейной конвертации RGB в Grayscale:

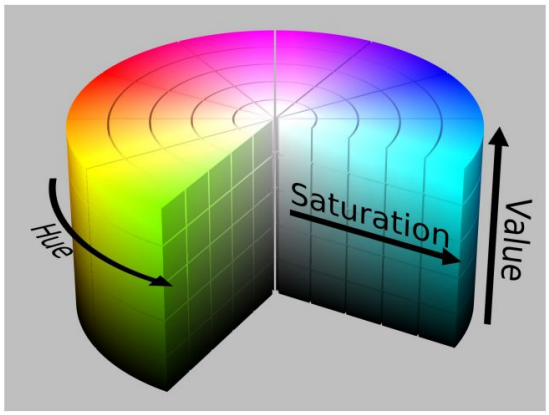


$$S = 0.2126R + 0.7152G + 0.0722B$$

Коэффициенты расставлены от того, как человеческий глаз воспринимает различные цвета.

HSV

HSV необходим для более интуитивного задания цвета человеку

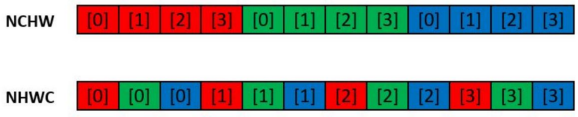


HSV цилиндр от SharkD, CC BY-SA 3.0

Hue – оттенок, Saturation – насыщенность, Value – яркость.

Когда говорят про растровые картинки важно знать каким образом хранятся цвета. Как правило, бывает два вида хранения картинок:
В питоновском формате PIL используется NHWC. В PyTorch взяли NCHW

Матричная репрезентация



CHW vs. HWC от Mingfeima

2 Croma key

https://colab.research.google.com/drive/1a6nOSheCdKkYq5ypEL-gUOpzfOCwykg2

Материал будет добавочным к блокам в colab

блок 1

PIL – библиотека, которая позволяет нам работать с картинками
numpy - позволяет работать с массивами
requsts – позволяет делать запросы по сети интернет
Из IO импортируем только BytesIO. BytesIO – позволяет открывать картинку из оперативной памяти
matplotlib – позволяет рисовать график (в данном colab, вроде, не нужен)

блок 2

Самая первая функция поможет нам получить картинку. Вторая функция засовывает картинку в массив чисел. Нормируем на 255, чтобы было легче работать. Третья функция массив превращает в картинку.

запуск трёх

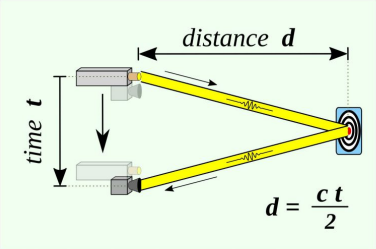
Третий блок просто рисует картинку.
Четвертый блок преобразует картинку в numpy array. Запускаем его. Измерения(250,590,3) – высота, ширина, каналы на RGB массив.
пятый блок просто отрезает картинку. Запускаем.
шестой, седьмой, восьмой блоки просто запускают каналы по отдельности. По ним видим сколько красного, зелёного, синего содержится в картинке. Белый – много , чёрный – мало.
Девятый блок убирает зелёный. Запускаем
Десятый блок убирает фон. Сам человек не идеально показывается и фон плохо убирается. Итог ужасный. Запускаем
Одиннадцатый блок нашу картинку преобразует в HSV. С помощью HSV лучше получается убирать задний фон. Можно сказать: Если пиксель зеленоватый, то это фон, если нет, то это человек.
Запускаем 12, 13, 14 блоки. (в них убираем Hue, Saturation, Value по очереди)
15 блок строит гистограмму для Hue. Из неё видим, что есть пик.
16 блок убирает пик
17 Остаётся что-то адекватное
18 блок берёт картинку
19 блок объединяет картинки
Запускаем
Видим, что на 19 блоке у картинки хромакей убран не совсем хорошо. Это вызвано тем, что зеленоватый оттенок передаётся волосам и волосы тоже убираются. Как это решить обсудим дальше.

3 Камеры глубины

Существует **Time of Flight камера.**

Лампочка моргает, свет доходит до объекта и отражается. Отражённый свет камера вылавливает и можно таким образом измерить на каком расстоянии от нас находится объект. Объекты, которые находятся ближе к нам дадут "пик"отраженного света быстрее, чем удалённые, из пиков во времени происходит определение отдаленности от нас. Могут быть проблемы с бликами от солнца. Если камера движется, то сложновато найти расстояние.

Time of Flight

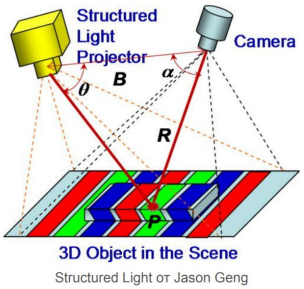


ToF от RCraig09, CC BY-SA 4.0

Stuctured Light.

Используется в айфонах.
Светит на объект, а потом камера вылавливает. Зная углы под которым находятся камера и излучитель света, можно определить на каком расстоянии находится объект. На краях объекта возникают проблемы.

Structured Light



Structured Light от Jason Geng

https://colab.research.google.com/drive/1vmRdiFMuySCCJuuVPHYxPzUDpESg1Rxs?usp=sharing
Материал будет добавочным к блокам в colab
смотрим как работает камера глубины, как это помогает отделить фон.
Лектор скачал демку с интернета, которую тоже будем использовать.
первый блок – импорт модулей
второй блок – то же, что было
третий блок – лектор

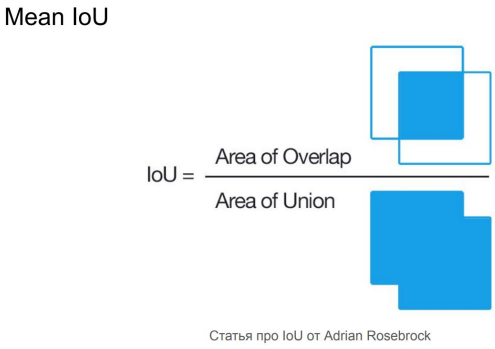
четвёртый блок – страшные преобразования из-за того, что автор демки не хотел морочиться с кодом и из-за того, что файлы хранились в неудобном формате для работы. Лектор находился близко и он чёрный, а фон далеко и он светлый.
пятый блок – построение гистограммы по глубине. Есть кластер близкий к камере – лектор. Далеко – штора.
шестой блок – что-то получаем с чем-то далёким и близким. Волосы получились ужасными, ибо камера не смогла уловить свет. На волосах сложная структура.
седьмой блок – убираем просто фон.
восьмой блок – интересные методы. inpaint старается заместить пиксели, которые попросим. Постараемся угадать какой был цвет у этих пикселей..
девятый блок. В метод inpaint суем массив через глубины и скажем подсунь нужные значения, где у нас NaN. Идея jointBilateralFilter в том, чтобы сглаживать картинку. В данном случае сглаживаем глубину и используем изначальные значения RGB картинки для того, чтобы это сделать. Алгоритм: смотрит значения вокруг пикселя и пытается для его значения усреднить те пиксели, у которых похожие RGB значения. Но с футболкой имеются проблемы из-за фона. Глубина помогает улучшить границу удаления фона.
11 блок – я не понял.
12 блок – я не понял.
Как я понял в конечных блоках удаляем границы с применением RGB и глубины и получается что-то дельное.

4 Сегментация изображений

Для сегментаций изображений используют нейронки **Pascal VOC** Датасет Pascal VOC является одним из бенчмарков, на котором часто меряют эффективность сегментации нейросети. В нём есть различные классы, нотации для классификации, детекции и сегментации
Нас интересует в данном разделе сегментация.
Перед сегментации изображений стоит задача примерно такая:
Каждому пикселю картинки соответсвует label class. Например, если у нас есть пиксель "фара автобуса поэтому он будет соответсвовать классу "автомобиль". Есть пиксель дороги, для которого своей нотации нет, поэтому он будет иметь класс "other".
В Pascal VOC есть класс для человека, им мы воспользуемся.

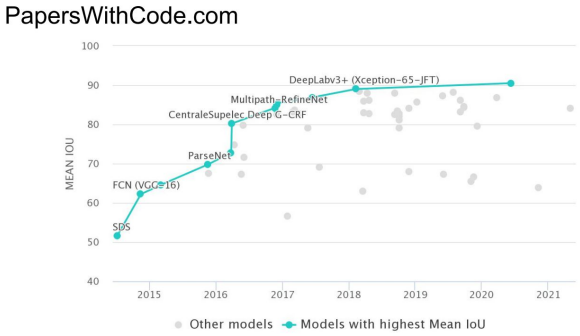
Определение эффективности алгоритма для классификации

Самая классическая **метрика Mean IoU**
Суть метрики: площадь правильных ответов делится на площадь неправильных ответов:



Смотри на картинку выше
В числителе площадь пересечения всех точек с правильными ответами, а в знаменателе площадь всего рисунка.
Важно, чтобы все точки с правильными ответами пересекались.

Самые эффективные относительно Mean IOU методы:



Сейчас будет рассмотрена нейронка
DeppLabv3+
https://colab.research.google.com/drive/1iN3ih0JTpC4C27P-9FGzrcmMCckE3SP2?usp=sharing
Первые три блока, как в предыдущих колабах
Четвертый блок – для DeepLabv3+ есть удобный способ с torch. В этом случае воспользуемся фреймворком transform(из популярных ещё есть tensorflow). Качаем модель с torch. Переключаем в eval режим(работает не на обучение).
Пятый блок – преобразовываем изначальные картинки в удобный формат для чтения для torch. Прогоняем нейронку на нашем входе
Шестой блок – смотрим для 15-го класса(для людей) сигмационную модель.
В дальнейших блоках мы убираем фон и подставляем новый.
Особенность сегметационных моделей – края.Это происходит из-за того, что сегментационные модели сильно шрафуют, если они сильно промахиваются от истинного значения.

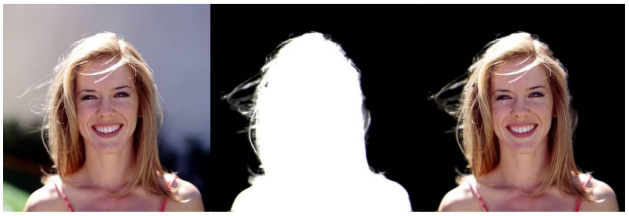
5 Маттинг изображений

Маттинг изображений – самое удачное решение для замены фона.
С маттингом мы не только хотим угадать к переднему плану изображения или к заднему плану относится тот или иной пиксель, но и нас интересуют дополнительные вещи:
В картинке могут быть полупрозрачные предметы, к примеру, волосы. Через волосы просвечивается фон. Нам важно узнать то насколько большую долю несёт передний план в этой смеси, и восстановить исходное значение для того, какой цвет был у фона. К примеру, на хромакее волосы были зеленоватыми, потому что фон передавал свой цвет. Но если узнать изначальное значение цвета фона, степень прозрачности и после этого подставить новые пиксели в фон, то волосы будут практически незменными после удаления фона.
На каринке выше показан пример использования маттинга.
https://colab.research.google.com/drive/1nbTt7ustaLc1MGAIrn7AGexla34Ix1Co?usp=sharing
Блоки не сильно отличаются от тех, что были до этого, поэтому я решил их не комментировать

Маттинг

Смесь полупрозрачных объектов

$$I^i = \alpha^i F^i + (1 - \alpha^i) B^i$$



Пример маттинга из Chuang et al

6 Что дальше?

https://peterl1n.github.io/RobustVideoMatting//
https://omnimatte.github.io/
https://augmentedperception.github.io/total_relighting/