Конспект по курсу «Компьютерная графика»

Программная инженерия. 3 курс.

Лектор: В. Гориховский

Санкт–Петербург
2023

Содержание

Введение от лектора

Компьютерная графика – крайне обширная область всесторонне занимающаяся работой с графической информацией. В курсе будут рассмотрены лишь некоторые её разделы. На лекциях мы будем с разных точек зрения изучать графику: математически, алгоритмически и инструментально. К каждому разделу курса прилагается лабораторная работа. Успешное выполнение повышает максимальную оценку, которую можно получить на зачёте:

Зачтено лабораторных	ECST
0	E
1	D
2	C-B
3	B-A

Таблица 1: Зависимость максимальной оценки от числа выполненных лабораторных.

1-2 человека из группы, кто справились с лабораторными особо выдающимся образом (сугубо субъективно), а также авторы конспекта по одной лекции, получают бонус "+1"к итоговой оценке. Также на последних занятиях можно сделать доклад о разных инструментах. Успешное выступление оценивается на плюс балл для тех, кто сдал экзамен на не менее чем удовлетворительную оценку.

Первый раздел курса посвящен геометрическим основам работы с графической информацией: взаимодействие геометрических объектов на плоскости и в пространстве. На первых двух лекциях рассмотрим некоторые алгоритмические задачи вычислительной геометрии: построение выпуклых оболочек и триангуляций, задачи локализации точки и близости, а также начала геометрической классификации и Диаграммы Вороного. В качестве дополнительной литературы настоятельно рекомендую заглянуть в замечательную книгу [1].

1 Лекция 1: простейшие геометрические объекты и выпуклые оболочки.

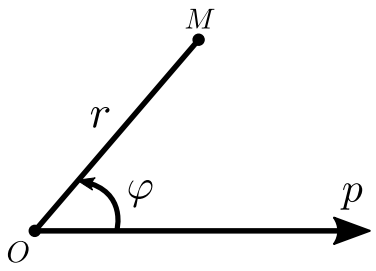
Конспект составила: *Кузьмина Ксения Антоновна*

Определение 1. Точка на плоскости – геометрический объект, задающийся минимум двумя координатами.

Системы координат, в которых мы можем задать точку на плоскости:

1. Афинные системы координат, в которой есть две произвольные оси координат и точка отсчета. И в частности, декартова система, состоящая из перпендикулярных оси абсцисс и оси ординат. Тогда точка задается координатами (x, y), где x – положение точки относительно оси абсцисс, а y – относительно оси ординат.

2. Полярная сестема координат. Выберем точку O и построим из неё луч p, который назовем *полярной осью*. Возьмем теперь произвольную точку M на плоскости и измерим две величины: расстояние от M до O и угол между отрезком OM и полярной осью. Обозначим расстояние за r, а угол за φ. Тогда, чтобы избежать неоднозначности, будем считать, что $r > 0, \varphi \in [0, 2\pi)$, и если $r = 0$, то $\varphi = 0$.



Определение 2. Геометрическая фигура — это множество точек на плоскости, ограниченных определенным числом линий. При этом допускаются вырождения, например прямая или луч.

Утверждение 1. Для однозначного задания прямой необходимы две точки.

Утверждение 2. Для однозначного задания фигуры на плоскости необходимы 3 точки. Иначе можно задавать фигуру с точностью до симметрии.

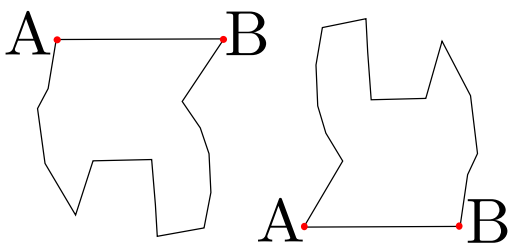


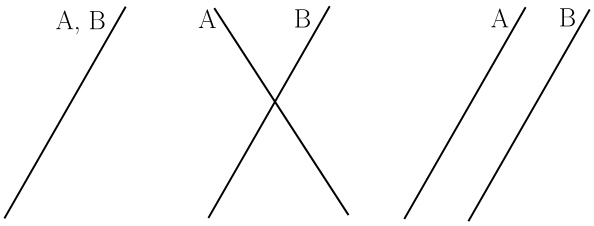
Рис. 1: Построение одной и той же фигуры по двум точкам

Утверждение 3. Прямую можно задать уравнением $Cx+Dy = E$, где C, D и E — постоянные коэффициенты, а все точки с координатами (x, y), удовлетворящими уравнению, принадлежат прямой.

Утверждение 4. Чтобы проверить лежит ли точка выше или ниже прямой, заданной уравнением $Cx + Dy = E$, можно рассмотреть неравенство $Cx + Dy < E$ или $Cx + Dy > E$.

Утверждение 5. Две прямые могут

- Совпадать
- Пересекаться
- Быть параллельными



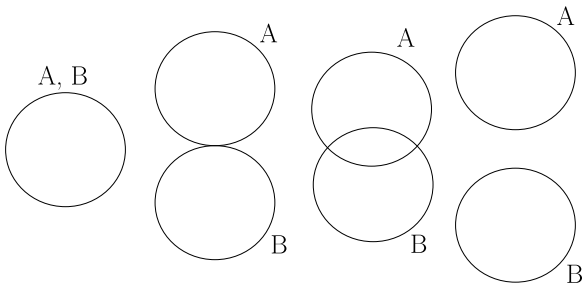
Определение 3. Окружность — геометрическая фигура, задаваемая уравнением

$(x-x_0)^2+(y-y_0)^2 = r^2$, где точка (x_0, y_0) — координаты центра окружности, а r — её радиус.

Утверждение 6. Окружности могут

- Совпадать
- Касаться
- Пересекаться
- Не пересекаться

Одним из подходов к объяснению того, что вариантов стало больше по сравнению с прямыми, может быть тот факт, что окружность задается квадратным уравнением.



Определение 4. Вектор — класс эквивалентности направленных отрезков.

Определение 5. Угол между векторами — величина, равная $\arccos \frac{(a,b)}{||a||\cdot||b||}$.

Определение 6. Выпуклая оболочка множества точек S — наименьшее выпуклое множество, содержащее S.

Задача 1. Построить выпуклую оболочку.

Алгоритм 1 (Алгоритм по определению). Оттолкнемся от идеи того, что отрезок является ребром выпуклой оболочки тогда и только тогда, когда все точки множества лежат по одну сторону от отрезка (либо на нем). n точек определяют $C_2^n = n^2$ прямых. Для каждой из прямых можно определить положение оставшихся $N - 2$ точек относительно рассматриваемого отрезка и понять, удовлетворяют ли они условию нахождения по одну сторону. Таким образом, итоговая сложность будет равна $O(n^3)$.

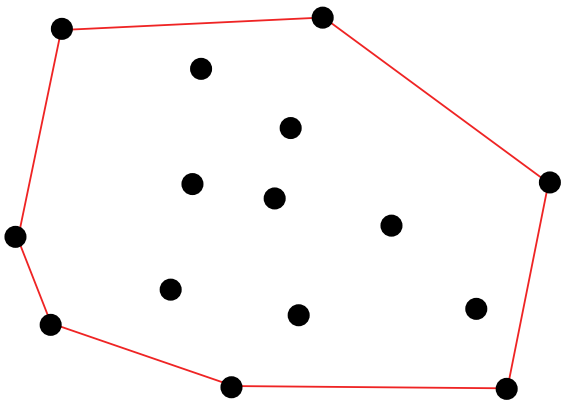


Рис. 2: Минимальная выпуклая оболочка для данного множества точек нарисована красной линией.

Алгоритм 2 (Алгоритм Джарвиса). Джарвис заметил, что если мы нашли отрезок ab, принадлежащий выпуклой оболочке, то тогда должно существовать ребро выпуклой оболочки, начало которой будет в вершине b.

Для начала возьмём точку p_1, которая точно принадлежит выпуклой оболочке, например, самую левую по координате x. Её можно найти за $O(n)$. Следующей точкой берём такую точку p_2, которая имеет наименьший положительный полярный угол относительно точки p_1 как начала координат. Аналогично точка p_3 имеет наименьший полярный угол относительно p_2 как начала координат. И каждая следующая точка выпуклой оболочки может быть найдена за линейное время.

В тот момент, когда следующая точка в выпуклой оболочке совпала с первой, алгоритм останавливается — выпуклая оболочка построена.

В каждой точке мы просматриваем остальные (кроме входящих в минимальную оболочку) на предмет минимальности полярного угла. Это выполняется за $O(n)$. Всего таких просмотров точек будет по количеству вершин в минимальной выпуклой оболочке, обозначим его за s. Таким образом, алгоритм работает за $O(s \cdot n)$. При малом количестве вершин в выпуклой оболочке алгоритм получается достаточно эффективным, но в худшем случае, когда все точки являются вершинами выпуклой оболочки, он отработает за $O(n^2)$.

Алгоритм 3 (Алгоритм Грэхема). Суть алгоритма Грэхема состоит в однократном просмотре упорядоченной последовательности точек, в процессе которого

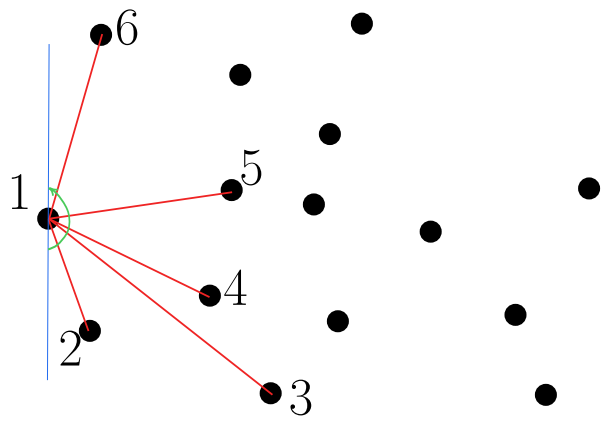


Рис. 3: Первый шаг алгоритма Джарвиса. Выбрали самую левую вершину и ищем наименьший угол между полярной осью и прямой, соединяющий две точки. Рисунок упрощен для лучшего восприятия, на самом деле проверка угла проводится для каждой вершины. Из рисунка видно, что наименьший угол будет между вершинами 1 и 2. Далее алгоритм повторится для точки 2, далее для точки 3 и так далее, пока обход не придет в точку 1.

удаляются внутренние точки. Оставшиеся точки являются вершинами выпуклой оболочки, представленными в требуемом порядке.

На первом шаге ищется любая точка в A, гарантированно входящая в MBO. Такой точкой, например, может быть точка с наименьшей x-координатой (самая левая точка в заданом множестве). Далее тройки последовательных отсортированных точек многократно проверяются в порядке обхода против часовой стрелки с целью определить образуют они угол больше или равный π. Если угол больше или равен π, то будем говорить, что точки образуют "правый поворот иначе назовем это "левым поворотом". А из выпуклости многоугольника будет следовать, что при обходе против часовой стрелки в нем могут быть только левые повороты.

После сортировки точек и выбора стартовой, необходимо пройти всё множество точек, выполняя следующие шаги:

- Если во время обхода множества у нас получился правый поворот на точках $p_1p_2p_3$, необходимо удалить из множества точку p_2 и рассмотреть $p_0p_1p_3$.
- Если $p_1p_2p_3$ образуют левый поворот, то мы продолжаем просмотр тройки $p_2p_3p_4$

Каждая проверка угла производится за постоянное время. После каждой про-

верки происходит либо продвижение на одну точку, либо удаляется точка. Так как множество содержит лишь n точек, то продвижение вперед может происходить не более n раз, как не может быть удалено и более n точек. Кроме того, необходимо потратить время на сортировку точек. Если взять сложность быстрой сортировки, равную $O(\log n)$, то получим итоговую сложность $O(n \log n)$.

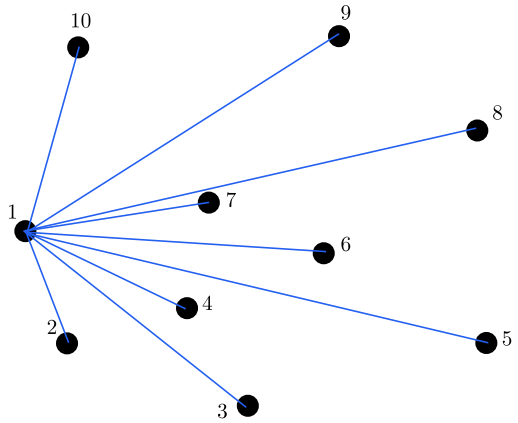


Рис. 4: Первый шаг алгоритма Грэхема, сортировка множества точек по углу. Далее обход будем проводить именно в таком порядке.

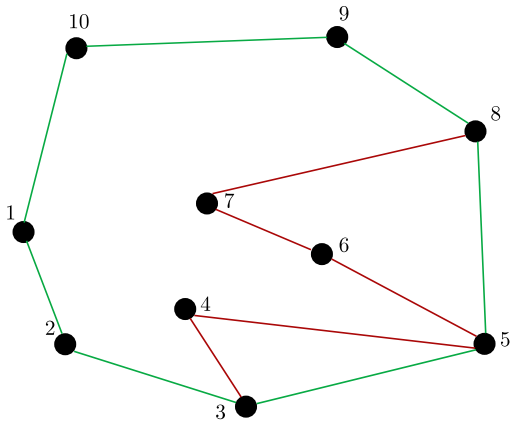


Рис. 5: Визуализация обхода алгоритма Грэхема. Красным обозначены пройденные, но непринятые пути, зеленым - пройденные и принятые.

Замечание. Алгоритм Грэхема работает только на плоскости и не подлежит распараллеливанию.

Алгоритм 4 (Разделяй и властвуй). «Разделяй и властвуй» — принцип разработки алгоритмов, заключающайся в рекурсивном разбиении решаемой задачи на две или более подзадачи того же типа, но меньшего размера, и комбинировании их решений для получения ответа к исходной задаче. Такие алгоритмы хорошо поддаются распаралелливанию.

Предположим при решении задачи построения выпуклой оболочки, исходное множество точек было разбито на две части S_1 и S_2, каждая из которых содержит половину точек исходного множества, если рекурсивным образом найдены $\mathrm{CH}(S_1)$ и $\mathrm{CH}(S_2)$ (CH = convex hull = выпуклая оболочка), то каковы дополнительные затраты, необходимые для построения $\mathrm{CH}(S_1 \cup S_2)$? Для ответа на вопрос можно воспользоваться следующим соотношением: $\mathrm{CH}(S_1 \cup S_2) = \mathrm{CH}(\mathrm{CH}\ (S_1) \cup \mathrm{CH}\ (S_2))$, тогда сложность алгоритма является решением реккуретного соотношения $T(N) \leqslant 2T(\frac{N}{2}) + f(N)$, где $f(N)$ - время построения выпуклой оболочки объединения двух выпуклых многоугольников, каждый из которых имеет около $\frac{N}{2}$ вершин.

При первом взгляде может показаться, что такой способ требует больших затрат, чем прямой способ построения выпуклой оболочки, решающее соображение состоит в том, что $\mathrm{CH}(S_1)$ и $\mathrm{CH}(S_2)$ - не просто неупорядоченные множества точек, а выпуклые многоугольники.

Пусть мы уже имеем опорные прямые P_1 и P_2.

1. Найдем некоторую внутреннюю точку p многоугольника P_1. Такая точка будет внутренней точкой $\mathrm{CH}(P_1 \cup P_2)$
2. Возможно два случая.
   - Точка p не является внутренней точкой многоугольника P_2. Проведем две опорные прямые (прямые, проходящие через некоторые вершины многоугольника таким образом, что все внутренние точки многоугольника лежат по одну сторону от данной опорной прямой) для многоугольника P_2, проходящие через точку p. Эти опорные прямые проходят через вершины v и u многоугольника P_2. Все точки внутри треугольника vpu не принадлежат границе выпуклой оболочки $\mathrm{CH}(P_1 \cup P_2)$. Все остальные точки упорядочиваем по полярному углу относительно точки p слиянием двух упорядоченных списков вершин за время $O(N_1) + O(N_2) = O(N)$, а затем применяем к полученному списку метод обхода Грэхема, требующий лишь линейное время.
   - Точка p является внутренней точкой многоугольника P_2. Упорядочиваем вершины обоих многоугольников относительно центра p, сливая два упорядоченных списка вершин P_1 и P_2 за $O(N)$.

3. Теперь к полученному списку вершин можно применить алгоритм Грэхема за исключением фазы сортировки точек по полярной координате, тогда он будет выполнен за линейное время.

   В сумме все три фазы алгоритма выполняются за время $O(N)$. Таким образом в соотношении $T(N) \leqslant 2T(\frac{N}{2}) + f(N)$, $f(N) = O(N)$. По мастер-теореме получается, что $T(N) = O(N \log N)$

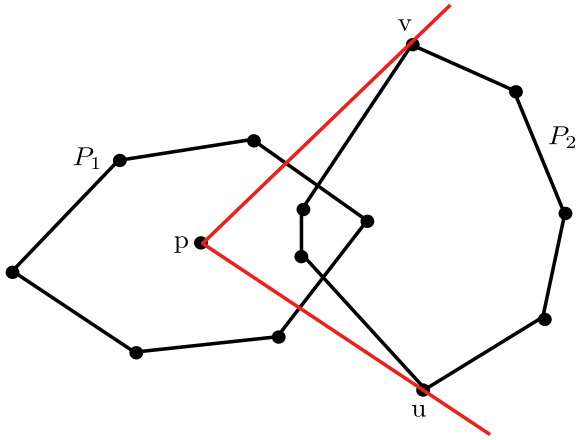


Рис. 6: Точка p находится вне многоугольника P_2. Многоугольник P_2 находится внутри угла, определяемого точками v, p и u. Точки v и u разбивают последовательность вершин многоугольника P_2 на две цепи. Одну из них можно удалить, а слияние вершин второй цепи с упорядоченным множеством вершин многоугольника P_1 можно выполнить за линейное время.

2 Лекция 2: Задача локализации точки и задача близости.

Конспект составил: ФИО

3 Лекция 3: Представление графической информации в памяти компьютера. Особенности хранения трехмерной информации

Конспект составил: *Печенев Данила Евгеньевич*

3.1 Восприятие трехмерного пространства

В прошлых лекциях мы обсуждали алгоритмы, которые позволяют решать те или иные задачи на плоскости. Поговорим теперь о трехмерном пространстве. Работать в нем сложнее, потому что мы воспринимаем мир объемным не напрямую, а путем достраивания до трехмерной модели плоских изображений, которые получают наши глаза. И не всегда эта модель соответствует действительности (простой пример — 3D рисунки на асфальте).

Восприятие трехмерного мира через двумерные изображения имеет ряд интересных особенностей. Примером может служить отношение ближе-дальше. Чем большим мы видим объект, тем ближе он к нам.

При этом важно отметить, что модели, которые мы храним в памяти компьютера, являются приближенными. Это вполне естественно: мы не можем хранить в ограниченной памяти континуальный объект (пример – число π). Это соображение переносится и на хранение изображений. Так появляется идея хранить и отображать изображения в пикселях.

3.2 Растровое представление изображений в памяти компьютера

Определение 7. Пиксель – это, с одной стороны, наименьший логический элемент двумерного растрового изображения, а с другой стороны, физический элемент матрицы дисплеев, отображающих изображение. Пиксель представляет собой неделимый объект прямоугольной формы, характеризуемый определённым цветом.

Определение 8. Растровое изображение – это изображение, представляющее собой сетку пикселей.

Характеристики пикселя:

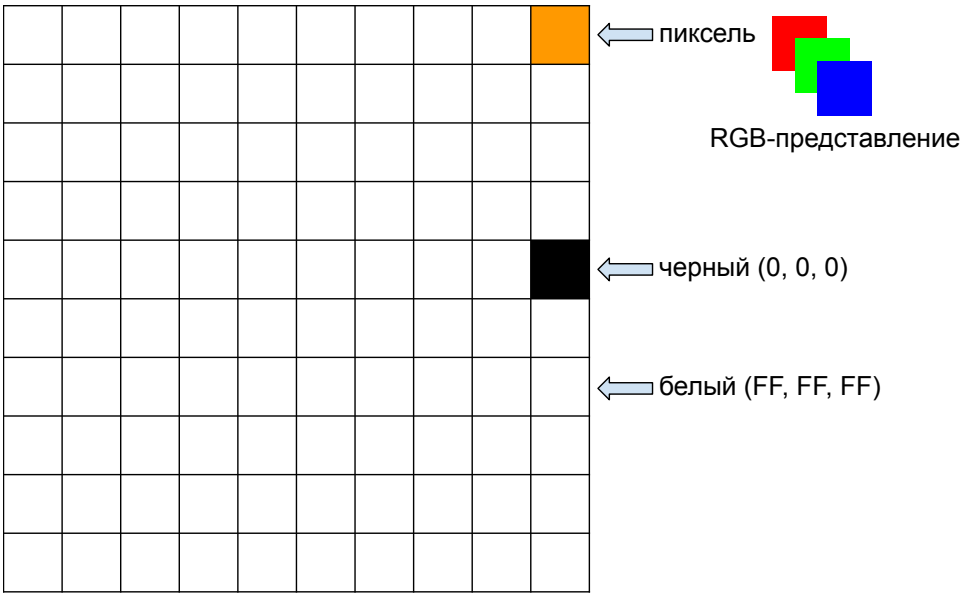


Рис. 7: Сетка пикселей. RGB-представление цвета пикселя. Черный и белый цвета в RGB-представлении.

- Цвет
- Яркость
- Прозрачность

3.3 Представление цвета

Как мы уже обсудили ранее, отображать цвета непрерывно не представляется возможным. Более того, это и не нужно. Человеческий глаз способен различить лишь небольшое количество цветов. Поэтому представление цвета дискретно.

Существуют разные модели представления цвета. Все они основаны на такой идее: все цвета можно получить комбинацией лишь нескольких базовых цветов. Самая известная и общепринятая модель — это RGB (Red, Green, Blue). В ней каждый цвет представляется как комбинация оттенков красного, зеленого и синего. Выбор основных цветов обусловлен особенностями физиологии восприятия цвета сетчаткой человеческого глаза.

Каждый из базовых цветов имеет 256 оттенков и представляется числом от 0 до 255, т.е. его хранение занимает 1 байт.

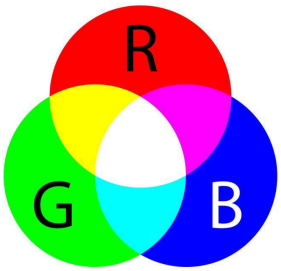


Рис. 8: RGB-модель представления цвета.

Преимуществом модели RGB также является то, что в неё естественным образом ложатся черный и белый цвета. На рисунке 7 показано их представление.

Другая модель представления цвета — это CMYK. В ней используются циановый, пурпурный, желтый и черный цвета: Cyan, Magenta, Yellow, Key. Key — это другое название черного.

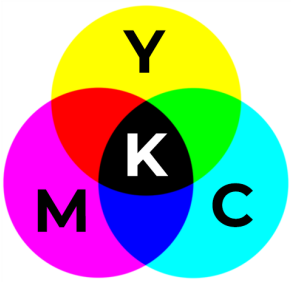


Рис. 9: CMYK-модель представления цвета.

(0, 0, 0) в модели CMYK — это белый цвет. И это отлично подходит для печати в издательствах, ведь изображения там печатаются на белой бумаге. Добавить к белому цвет в системе CMYK получается дешевле, чем в RGB (нужно меньше краски). Из цветов Cyan, Magenta и Yellow получить черный цвет нельзя. Именно поэтому в системе CMYK специально добавляется черный цвет.

3.4 Векторное представление изображений в памяти компьютера

Изображение разбивается на примитивы (элементы, фигуры). Благодаря этому изображение можно масштабировать без потери качества.

К недостаткам относятся:

- Сложность снятия векторного изображения
- Нетривиальное хранение
- Сложность отображения: прорисовка каждого векторного объекта, когда экран по-прежнему пиксельный

Основные форматы векторных изображений:

- .svg
- .pdf
- .eps

3.5 Сжатие изображений

Одним из самых популярных разрешений монитора является 1920x1080 (также известное как 1080p или Full HD). На экране такого монитора отображается в общей сложности 2 073 600 пикселей. Если каждый из них представляется хотя бы 3 байтами (RGB), то размер изображения уже составляет 6MB. И это вообще-то много! Изображения необходимо сжимать. Для этого существуют разные алгоритмы и преобразования. Как пример — вейвлетное преобразование, которое используется в графическом формате JPEG 2000.

Алгоритмы сжатия чаще всего предназначены для конкретного типа информации. Для изображений подходят одни, для аудио другие и так далее. Поэтому такие общие алгоритмы, как, например, алгоритм Хаффмана, на изображениях могут работать очень плохо.

Конкретно к изображениям может быть применена следующая идея. Разница в значениях цветов соседних пикселей чаще всего будет минимальной. А это значит,

что мы можем сгруппировать часть пикселей и хранить их более оптимально. Более того, если нас устраивает небольшая потеря качества, можно группировать пиксели с очень похожими цветами и говорить, что там один и тот же цвет. Качество упадет несущественно, зато можно значительно сэкономить используемую память.

3.6 Сжатие видео

Видео представляет собой последовательность кадров (изображений). Если хранить их «в лоб», размер одного фильма может достигать нескольких сотен гигабайт! Поэтому хранение видеоряда необходимо сильно оптимизировать.

Здесь применяется следующая идея. Для абсолютного большинства кадров разница между подряд идущими кадрами будет крайне мала. Эту разницу, т.е. набор изменившихся при переходе на следующий кадр пикселей, мы и будем хранить.

3.7 Хранение трехмерной информации

Трехмерные модели:

- Воксельные. Трехмерная сетка, вместо пикселей воксели — трехмерные элементы сетки. Подходят для 3D моделирования. Многие вычислительные методы используют именно сетки. Недостаток: хранение большого числа лишней информации, ведь нас часто интересует только поверхность объемного объекта, а не его внутренность или окружение.

- Аналитические. Недостаток: очень сложно задавать объемные объекты аналитически.

- Полигональные. Если три или более точек координат заданы в качестве вершин и соединены рёбрами, то они формируют многоугольник (полигон), который может иметь цвет и текстуру. Соединение группы таких полигонов позволяет смоделировать практически любой объект. Недостаток: все объекты должны состоять из крошечных плоских поверхностей, а полигоны должны иметь очень малый размер, иначе края объекта будут иметь огранённый вид. Это означает, что если для моделируемого объекта предполагается увеличение, его

необходимо моделировать с большим количеством полигонов (плотностью), даже несмотря на то, что большинство из них будут лишними при удалении от объекта.

Часто эти модели можно совмещать, т.е. использовать вместе.

4 Лекция 4: Реализм в компьютерной графике. Физичность и кинематографичность модели.

Конспект составил: *Алексеев Артур Игоревич*

Утверждение 7. Реализм модели можно рассматривать в двух подходах:

- Физичность
- Кинематографичность

Определение 9. Физическая модель - абстракция реальных объектов или систем в трехмерном пространстве. Эта модель учитывает физические свойства объекта, такие как форма, размер, текстуры, и материалы, чтобы создать визуальное восприятие, близкое к реальности.

Определение 10. Кинематографическая модель - модель, которая фокусируется на движении и анимации объектов и не стремится правдоподобно изобразить реальность. Для такой модели главное - красивый кадр.

Модели с точки зрения физики

Утверждение 8. При попытке расчета деформаций для всей модели, мы сталкиваемся с серьезными вычислительными сложностями. Прежде всего, это связано с обширностью данных и высокой степенью свободы, которые представляют собой деформации в трехмерном пространстве. Рассчитывать каждую точку модели в каждый момент времени требует огромных вычислительных ресурсов.

Замечание. На основе вышеуказанного утверждения, дальнейшие рассуждения будут проводиться на основе моделей, которые могут **только изгибаться** . Растяжения и поломки пока что не рассматриваем.

Определение 11. Сеточная модель - аппроксимация геометрического объекта, посредством разбиения его на конечное количество элементов, образующих сетку. Эта сетка состоит из вершин, рёбер и граней, которые приближенно представляют форму и структуру оригинального объекта.

Основные компоненты:

- Вершины: Точки, определяющие узлы сеточных элементов.
- Рёбра: Соединяют вершины и формируют грани элементов.
- Грани: Плоские поверхности между рёбрами, ограничивающие объем элемента.
- Элементы: Малые части объекта, образованные вершинами, рёбрами и гранями.

Метод 1. Метод конечных элементов (МКЭ) представляет собой численный метод для решения уравнений, описывающих поведение объектов под воздействием внешних сил. В компьютерной графике, он часто используется для анализа деформаций.

Идея:

- **Дискретизация:** Объект разбивается на конечные элементы. Это может быть треугольная, квадратная сетка и так далее, в зависимости от формы объекта. Кроме того, сетка может быть неравномерной для лучшего моделирования.
- **Определение уравнений:** Для каждого элемента составляются уравнения, описывающие его деформации. Эти уравнения обычно включают дифференциальные уравнения, связанные с физическими свойствами материала.
- **Сборка матриц:** Полученные уравнения объединяются в систему линейных уравнений. Это создает матрицу, описывающую всю систему.
- **Решение системы уравнений:** Применяются численные методы для решения системы уравнений.
- **Вычисление деформаций:** После решения системы уравнений можно вычислить деформации и напряжения в каждом элементе. Это позволяет получить представление о поведении объекта под воздействием нагрузок.

Замечание. В процессе моделирования сетка может оказаться состоящей из миллионов ячеек, каждую моделируют десятки уравнений. У каждого уравнения копится погрешность.

Кроме того, для решения уравнений можем использовать метод Гаусса с алгоритмической сложностью $O(n^3)$.

$10^6 \cdot 10 = 10^7$ - размер матрицы миллиона ячеек, у каждой из которых есть по десять уравнений. Итого, применяя метод Гаусса, получаем следующее итоговое время вычисления: $(10^7)^3 = 10^{21}$. Даже если учесть мощность современного процессора в несколько гигагерц, получим, что на просчет одного кадра потребуется около 10^{12} тактов, что может занимать несколько лет. Так, например, на моделирование самолета Boeing уходит около 30 месяцев.

Вычисления можно ускорить, так как в процессе получаются достаточно разреженные матрицы, но они все ещё достаточно плохо параллелятся.

Определение 12. **Шарнирная (скелетная) модель** - модель соединения, позволяющая моделируемым объектам изменяться только в точках сочленения. Предполагается, что каждая вершина модели «привязывается» к какой-либо кости скелета. Таким образом, при движении отдельной кости двигаются и все вершины, привязанные к ней, что упрощает расчеты.

Определение 13. **Скелет** - многоугольник, задающий модель, который подвергается преобразованиям.

Утверждение 9. Для выпуклых многогранников можно найти скелет за $O(n^2)$ При учете того, что модель гнется только по точкам сочленения, преобразование скелета - линейное преобразование прямых.

Когда стороны движутся, сжимаясь параллельно себе с постоянной скоростью, вершины, где пары сторон пересекаются, также движутся со скоростью, зависящей от угла при вершине. Если одна из этих движущихся вершин сталкивается с несмежной стороной, многоугольник распадается на два и процесс продолжается для каждой части отдельно.

Если траектории движения вершин скелета пересекаются в какой-то момент времени, это означает, что сторона скелета, образуемая двумя этими вершинами, "схлопывается".

Утверждение 10. На скелетных моделях можно основывать физические модели. Для этого:

1. Задаем массы на шарниры и конечные точки (для гравитации)
2. Используем закон Гука на кости (Mass-Spring), в итоге кости смоделированы как пружины.

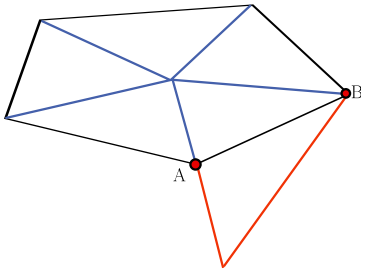


Рис. 10: Черным цветом обозначена фигура, синим - её скелет. Допустим, точки A и B движутся по красным траекториям. Когда они пересекутся, сторона AB выродится.

Утверждение 11. Бывают материалы которые растягиваются по-разному в разные стороны, например человеческая кожа (тянется во все четыре стороны, но не вверх/вниз).

Используя моделирование кожи можно понять, с какого участка и сколько брать, чтобы она прижилась после пересадки, например, для лечения ожогов, однако просто моделирования скелета может быть недостаточно.

Утверждение 12. Подход Mass-tensor вводит тензорное взаимодействие. Таким образом, тензоры могут хранить не только вышеперечисленные взаимодействия, но, например, и скручивания.

Утверждение 13. Реалистичное физичное моделирование жидкостей гораздо более трудоёмкий процесс, чем моделирование твердых тел. Например, уравнения Навье-Стокса, описывающие движение жидкости, представляют собой систему нелинейных уравнений, что существенно усложняет их решение по сравнению с уравнениями, описывающими твердые тела.

Даже если сильно упрощать модель, в играх, скорее всего, мы сможем увидеть не более половины кадра в секунду, поэтому по факту чаще всего используются текстуры и предпосчитанные эффекты.

5 [WIP] Лекция 5: Компьютерное зрение

Конспект составил: *Бучин Вячеслав Андреевич*

TODO:Здесь нужно привести много примеров

В области компьютерного зрения выделяют следующие задачи.

1. Детекция ($\delta : \{Image\} \to \{0, 1\}$) — проверить, удовлетворяет ли данное изображение какому-либо свойству.

2. Классификация ($k : \{Image\} \to \{0, 1, \ldots, N\}$) — определить, какой группе из заранее заданных относится данное изображение.

3. Сегментация ($s : \{Image\} \to \{Image\}^n$)— некоторым образом выделить из изображения определенные части, обладающие каким-то свойством.

Эти задачи сводимы друг к другу. Рассмотрим примеры в общих чертах.

- Детекция $\to$ классификация. Имея N подходящих под классы классификации детекторов, можно запустить каждый из них на изображении. Класс одного из сработавших детекторов можно считать результатом классификации.

- Классификация $\to$ детекция. Имея классификатор, среди классов которого присутствует тот, который хочется задетектировать, можно запустить классификатор, если

6 Лекция 7: Машинное обучение в Computer Vision

Конспект составил: *Задорожный Александр Сергеевич*

6.1 Traditional Approach vs ML

В настоящее время подходы с использованием машинного обучения преобладают в Computer Vision по нескольким причинам:

1. Эффективность: машинное обучение позволяет создавать модели, которые превосходят традиционные подходы на сложных задачах, таких как распознавание объектов, обнаружение лиц и классификация изображений.

2. Универсальность: машинное обучение может применяться к широкому спектру задач Computer Vision, что делает его более гибким, чем традиционные подходы.

3. Простота использования: машинное обучение можно использовать без глубокого понимания принципов Computer Vision, что делает его более доступным для широкого круга пользователей.

Традиционные подходы к Computer Vision, такие как анализ изображений и обработка изображений, в основном основаны на ручном программировании правил и алгоритмов. Эти подходы могут быть эффективными для простых задач, но они часто оказываются неэффективными для сложных задач, таких как распознавание объектов в сложных условиях.

Машинное обучение, с другой стороны, позволяет создавать модели, которые могут обучаться на больших наборах данных изображений. Эти модели могут научиться распознавать объекты и другие объекты в сложных условиях, которые были бы невозможны для традиционных подходов.

> **Замечание.** Стоит отметить, профессионалы, которые умеют в ручной Computer Vision ценятся сильно выше чем просто ml инженеры. Так как они скорее всего тоже умеют в ml, однако предпочитают ручные способы из-за высокой производительности.

Вот несколько конкретных примеров того, как машинное обучение используется в Computer Vision:

1. Распознавание объектов: машинное обучение используется для создания моделей, которые могут распознавать объекты на изображениях. Эти модели используются в широком спектре приложений, включая самоуправляемые автомобили, системы безопасности и системы распознавания лиц.

2. Обнаружение лиц: машинное обучение используется для создания моделей, которые могут обнаруживать лица на изображениях. Эти модели используются в приложениях, таких как потоковое видео и системы безопасности.

3. Классификация изображений: машинное обучение используется для создания моделей, которые могут классифицировать изображения по категориям. Эти модели используются в приложениях, таких как поиск изображений и анализ изображений.

6.2 Построение синтетических данных

Синтетические данные играют важную роль в Computer Vision, поскольку они могут использоваться для решения ряда проблем, связанных с реальными данными.

6.2.1 Важность синтетических данных в Computer Vision

Определение 14. Синтетические данные - это данные, которые были созданы искусственным образом, а не получены из реального мира.

Они могут быть полезны в Computer Vision по нескольким причинам:

1. **Увеличение размера набора данных:** синтетические данные могут использоваться для увеличения размера набора данных для обучения моделей машинного обучения. Это может быть полезно для задач, для которых трудно или дорого собирать реальные данные. Например, рассмотрим задачу распознавания поражения кожи. Это задача, для которой очень мало реальных данных, поскольку получение данных о поражениях кожи является дорогостоящим и трудоемким процессом. Синтетические данные могут быть использованы для

создания большего количества данных для обучения модели распознавания поражений кожи. Это может привести к улучшению точности модели и сделать ее более доступной для использования в реальных приложениях.

2. **Получение данных из недоступных источников:** синтетические данные могут быть использованы для получения данных из источников, которые недоступны для сбора реальных данных. Например, для обучения модели распознавания объектов на спутниковых изображениях может быть трудно или невозможно получить доступ к реальным изображениям.

3. **Получение данных с желаемыми свойствами:** синтетические данные могут быть созданы с желаемыми свойствами. Например, для обучения модели распознавания лиц в условиях низкой освещенности могут быть созданы синтетические изображения лиц с низкой освещенностью. Это может помочь улучшить точность модели в реальных условиях.

Рассмотрим более подробно две актуальные проблемы, касающиеся реальных данных:

- **Слишком большое количество данных**. В некоторых случаях может быть доступно слишком большое количество реальных данных. Это может привести к нескольким проблемам:

    1. Коллапс модели: модель может быть переобучена на слишком большом количестве данных. Это означает, что модель будет слишком хорошо подходить к обучающему набору данных, но будет плохо работать на новых данных.
    2. Затраты на хранение и обработку: хранение и обработка больших наборов данных может быть дорогостоящей.
    3. Время обучения: обучение модели на больших наборах данных может занять много времени.

    Рассмотрим задачу классификации изображений автомобилей. В настоящее время доступно огромное количество данных о автомобилях, включая изображения, видео и данные о характеристиках автомобилей. Это количество данных

может быть слишком большим для обработки и обучения модели, что может привести к переобучению модели. Синтетические данные могут быть использованы для сокращения размера набора данных, не теряя при этом важной информации.

- **Коммерчески закрытые данные** В некоторых случаях данные могут быть коммерчески закрытыми. Например, Google может иметь данные о поисковых запросах, но не может их раскрыть общественности. В таких случаях синтетические данные могут быть использованы для создания синтетических данных, которые похожи на реальные данные. Это может помочь исследователям и разработчикам улучшить свои алгоритмы без доступа к реальным данным.

6.2.2 Что такое хорошие синтетические данные?

Определение 15. Хорошие синтетические данные должны быть похожи на реальные данные. Это означает, что они должны иметь правильные характеристики, такие как распределение пикселей, яркость, контрастность и т. д. Распределение синтетических данных должно соответствовать распределению реальных данных. Например, если реальные данные содержат 50% изображений мужчин и 50% изображений женщин, то синтетические данные также должны содержать 50% изображений мужчин и 50% изображений женщин.. Т.е хорошие синтетические данные должны быть: точными(отражать реальные данные), разнообразными(охватывать широкий спектр возможных условий), качественными(быть четкими и хорошо освещенными)

Существует несколько различных синтетических моделей процессов, которые могут использоваться для создания синтетических данных. Некоторые из наиболее распространенных моделей включают:

1. **Модели на основе физических законов:** эти модели используют физические законы для создания синтетических данных. Например, модель может использоваться для создания синтетических изображений объектов, перемещающихся в пространстве.

2. **Модели на основе машинного обучения:** эти модели используют методы машинного обучения для создания синтетических данных. Например, модель может использоваться для создания синтетических изображений, похожих на изображения из реального набора данных.

Таким образом, синтетические данные играют важную роль в Computer Vision. Они могут использоваться для решения ряда проблем, связанных с реальными данными, включая нехватку данных, сложность реальных данных, коммерческую закрытость данных и т.д.

6.3 ML подходы для распознавания изображений

Определение 16. Распознавание изображений — это задача машинного обучения, которая заключается в определении объектов или явлений на изображении. Это широко используемая задача в различных приложениях, таких как самоуправляемые автомобили, системы безопасности и анализ изображений.

Существует множество различных подходов к распознаванию изображений, основанных на машинном обучении. Некоторые из наиболее распространенных подходов будут рассмотрены ниже.

6.3.1 Convolutional Neural Networks

Определение 17. Convolutional Neural Networks (CNN) являются одним из самых эффективных подходов к распознаванию изображений. Они основаны на концепции свертки, которая является математической операцией, которая используется для обнаружения локальных закономерностей в данных. CNN состоят из нескольких слоев, каждый из которых выполняет определенную задачу.

Основные принципы работы CNN

Свертка: первый слой CNN выполняет свертку входного изображения с набором весов. Результат свертки называется картой признаков. Карты признаков представляют собой набор признаков, которые были обнаружены в исходном изображении.

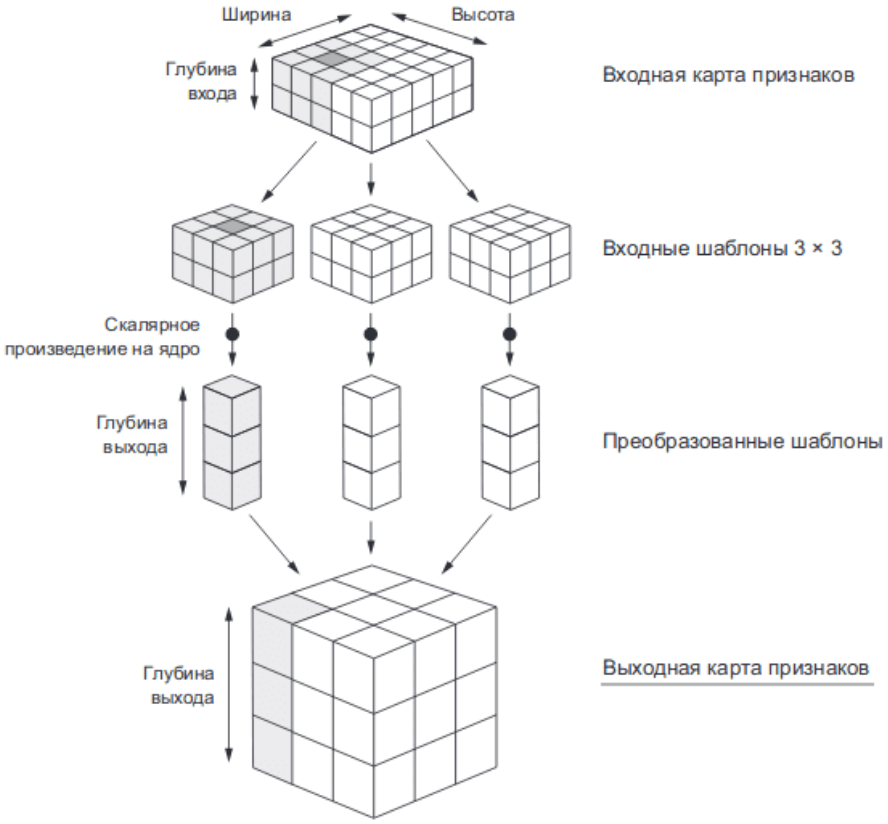


Рис. 11: Принцип действия операция свёртки.

Подвыборка: второй слой CNN выполняет подвыборку карт признаков. Подвыборка позволяет уменьшить размер карт признаков, что может улучшить производительность CNN.

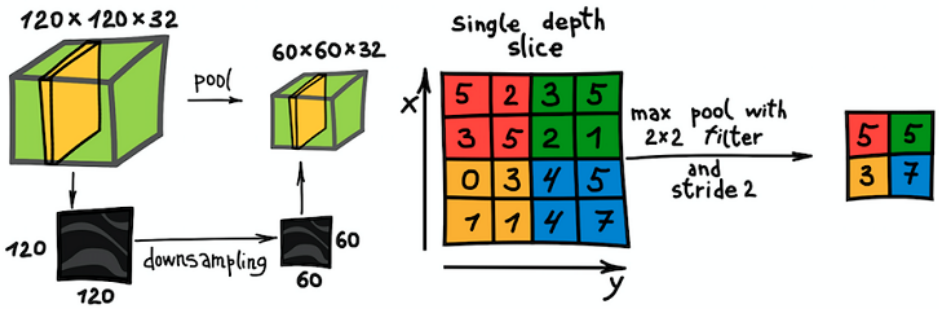


Рис. 12: Примеры подвыборки. (слева) Как подвыборка меняет пространственные (но не канальные!) размеры массивов данных, (справа) Принципиальная схема работы подвыборки

Полносвязный слой: последний слой CNN является полносвязным слоем. Полно-

связный слой принимает входные данные от карт признаков и выдает предсказание класса для входного изображения.

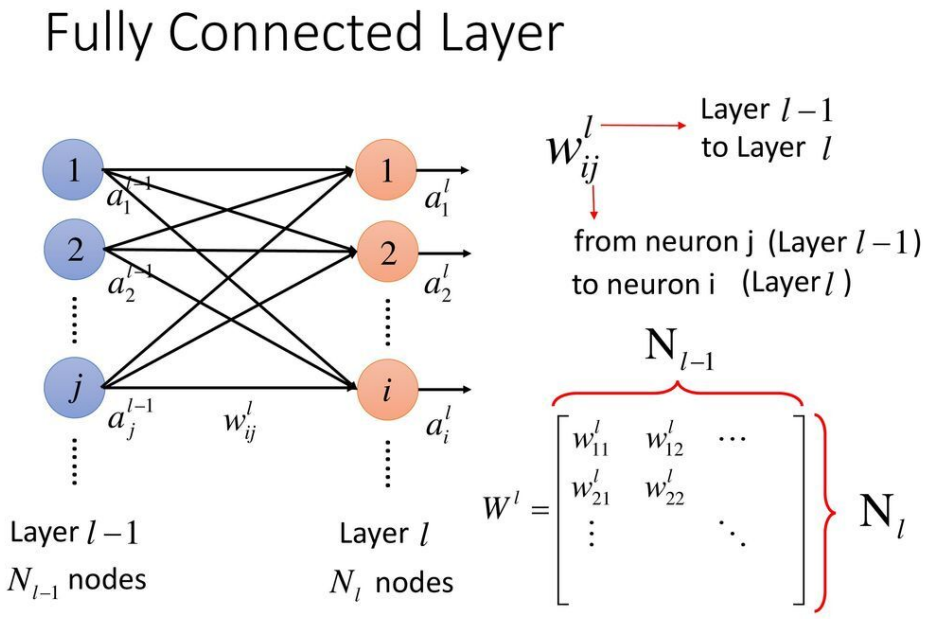


Рис. 13: Принцип действия полносвязного слоя.

CNN имеют следующие преимущества

1. **Эффективность:** CNN могут достигать высокой точности распознавания изображений.
2. **Универсальность:** CNN могут использоваться для решения широкого спектра задач распознавания изображений.
3. **Независимость от признаков:** CNN могут обнаруживать признаки изображений автоматически, что упрощает процесс разработки моделей распознавания изображений.

CNN также имеют следующие недостатки

1. **Зависимость от данных:** CNN требуют больших наборов данных для обучения.

2. **Черным ящик:** CNN состоят из большого количества сверточных ядер, которые выполняют сложные вычисления. Эти вычисления не могут быть легко поняты человеком, что затрудняет интерпретацию того, как CNN принимает решения. Существуют некоторые методы, которые могут помочь понять, как работают CNN. Например, можно использовать методы визуализации, чтобы увидеть, как сверточные ядра реагируют на различные изображения. Однако эти методы не всегда могут дать полное понимание того, как CNN принимают решения.

Примеры использования CNN в задачах распознавания

- CNN используются в Google Photos для автоматического распознавания объектов и лиц. Это позволяет пользователям легко находить изображения, которые они ищут.
- CNN используются в Facebook для распознавания лиц в фотографиях и видео. Это позволяет Facebook предлагать пользователям рекомендации друзей и другие персонализированные функции.

6.3.2 Transformers

Определение 18. Трансформеры - это тип нейронной сети, который был разработан для обработки последовательностей. Эти наборы сначала токенизируются, а затем передаются в трансформаторы. Трансформаторы добавляют внимание (квадратичная операция - вычисляет попарное внутреннее произведение между каждой парой токенизированных элементов. По мере увеличения количества элементов увеличивается и количество операций).

В настоящее время в задачах обработки естественного языка (NLP) преобразователи(transformers) стали архитектурой goto (например, BERT, GPT-3 и т. Д.). С другой стороны, использование трансформаторов в задачах компьютерного зрения все еще очень ограничено. Большинство исследователей напрямую используют сверточные слои или добавляют определенные блоки внимания вместе со сверточными блоками для приложений компьютерного зрения (таких как Xception, ResNet, EfficientNet, DenseNet, Inception и т. Д.). В статье о Vision Transformer (ViT) реализована чистая модель преобразователя, без необходимости использования сверточных

блоков, для последовательностей изображений для классификации изображений. В документе показано, как ViT14 может достичь лучших результатов, чем большинство современных сетей CNN, на различных наборах данных распознавания изображений при использовании значительно меньших вычислительных ресурсов.

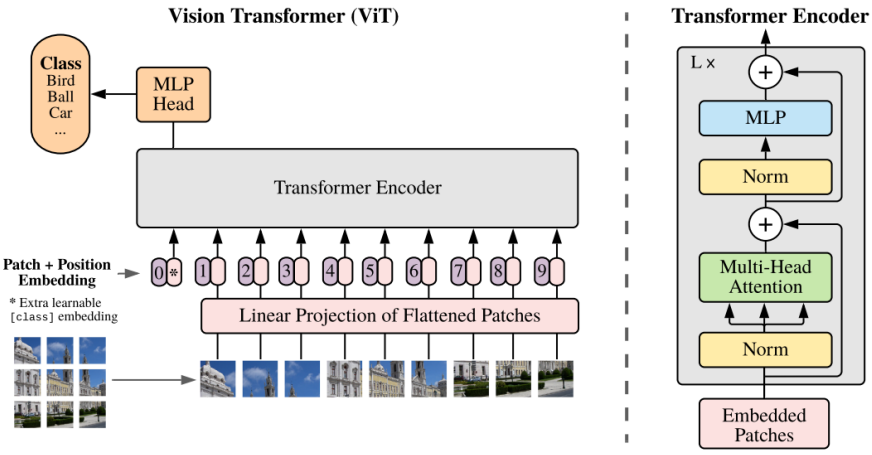


Рис. 14: Vision Transformer (ViT) Model

Преимущества трансформеров над CNN

- Трансформеры могут учитывать контекст всей последовательности при обработке каждого элемента последовательности. Это может быть полезно для задач распознавания изображений, где важно учитывать взаимосвязь между различными элементами изображения.
- Трансформеры могут быть более эффективными, чем CNN, для обработки последовательностей с длинными зависимостями.

До сих пор CNN руководили задачами компьютерного зрения. Изображение основано на идее, что один пиксель зависит от своих соседних пикселей, а следующий пиксель зависит от своих ближайших соседних пикселей (будь то цвет, яркость, контраст и т. Д.). CNN работает над этой идеей и использует фильтры на участке изображения для выделения важных деталей и краев. Это помогает модели узнавать

только необходимые важные особенности изображения, а не детали каждого пикселя изображения.

Если в модель загружаются данные всего изображения, а не только те части, которые фильтры могут извлечь (или они сочтут важными), шансы на то, что модель будет работать лучше, выше. Именно это и происходит внутри Visual Transformer. Это может быть одной из причин, почему в данном случае Vision Transformers работают лучше, чем большинство моделей CNN.

Однако это не означает, что в будущем трансформеры заменят CNN в задачах компьютерного зрения. Уже выпущена модель EfficientNet V2, которая работает лучше, чем Vision Transformers. Это означает, что теперь мы можем ожидать, что новые архитектуры обоих жанров (CNN и Transformers) будут бороться с этим, поскольку в ближайшем будущем будут появляться новые, лучшие и более эффективные модели.

6.3.3 Clustering

Определение 19. Кластеризация - это метод машинного обучения, который используется для группировки похожих объектов в группы, называемые кластерами.

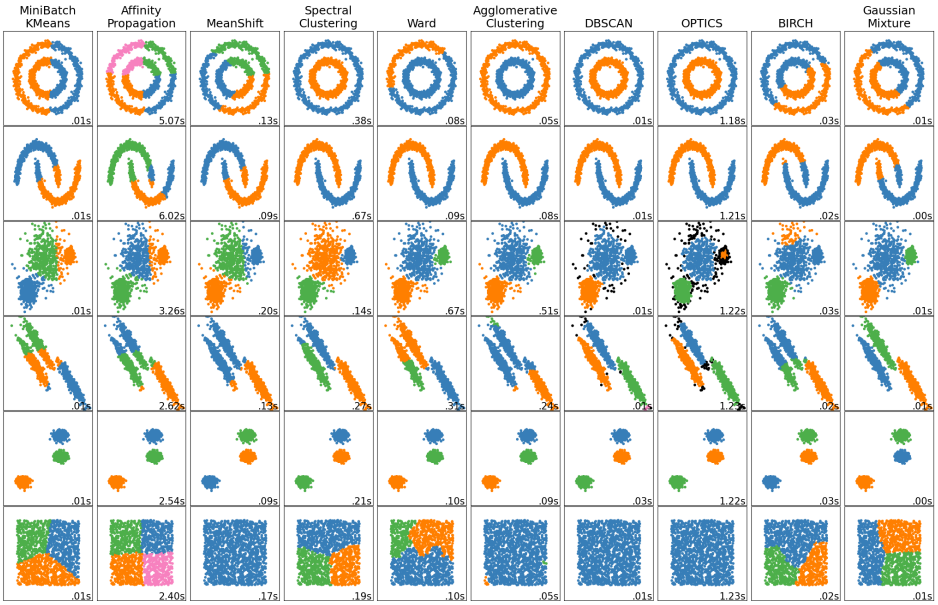


Рис. 15: Обзор методов кластеризации

Кластеризация может быть использована в задачах распознавания изображений

и классификации изображений для следующих целей:

- Предварительная обработка изображений: кластеризация может быть использована для предварительной обработки изображений для выделения областей интереса или для удаления шума.
- Повышение точности распознавания: кластеризация может быть использована для повышения точности распознавания изображений, разделяя изображения на более однородные кластеры.
- Расширение набора данных: кластеризация может быть использована для расширения набора данных для обучения моделей распознавания изображений, создавая новые кластеры из существующих изображений.
- Классификация объекта на изображении: рассмотрим задачу классификации изображений животных. Сначала мы можем предварительно обработать изображения, удалив шум и нормализовав их яркость. Затем мы можем использовать алгоритм кластеризации для группировки изображений в кластеры. После того, как кластеры будут созданы, мы можем назначить класс каждому кластеру. Например, мы можем присвоить класс «собака» кластеру, содержащему изображения собак.

Выбор типа кластеризации Выбор типа кластеризации зависит от конкретной задачи распознавания изображений или классификации изображений. Если необходимо получить кластеры с четко определенными границами, то целесообразным может быть использование центроидного кластеринга. Если необходимо получить кластеры с более размытыми границами, то целесообразным может быть использование группового кластеринга. Если необходимо учитывать динамику кластеров, то целесообразным может быть использование динамического кластеринга.

Кластеризация широко используется в задачах распознавания изображений и классификации изображений. Вот несколько примеров использования кластеризации в этих задачах:

- В самоуправляемых автомобилях кластеризация используется для обнаружения объектов на дороге. Объекты на дороге могут быть разделены на кластеры,

такие как автомобили, пешеходы и велосипеды. Это позволяет самоуправляемым автомобилям более эффективно обрабатывать информацию об окружающей среде.

- В системах безопасности кластеризация используется для обнаружения аномалий. Например, изображения людей могут быть разделены на кластеры, основанные на их внешнем виде. Если человек не соответствует ни одному из известных кластеров, то он может быть идентифицирован как потенциальный нарушитель.

- В системах распознавания лиц кластеризация используется для повышения точности распознавания. Изображения лиц могут быть разделены на кластеры, основанные на их сходстве. Это позволяет моделям распознавания лиц более точно определять лица, которые уже встречались ранее.

Кластеризация является мощным инструментом, который может быть использован в задачах распознавания изображений и классификации изображений для повышения эффективности и точности.

6.3.4 Random Forest

Определение 20. Random Forest - это алгоритм машинного обучения, который объединяет результаты множества деревьев решений. Деревья решений - это простые алгоритмы, которые принимают решение, разделяя пространство признаков на подпространства.

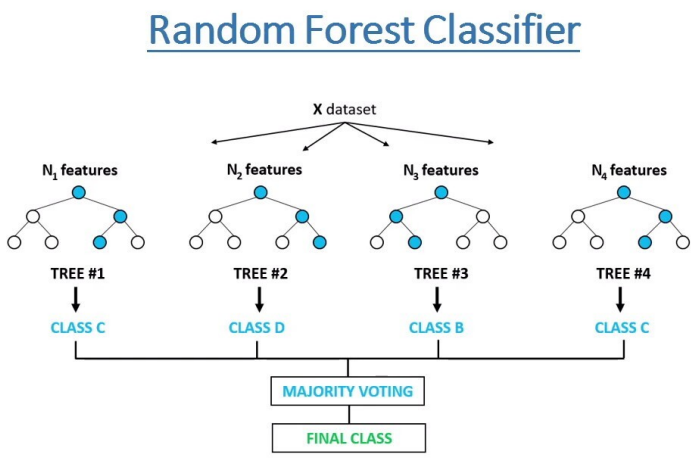


Рис. 16: Random Forest algorithm intuition

Как работает Random Forest?

Random Forest работает следующим образом. 1) Из обучающего набора данных генерируется множество деревьев решений. Каждое дерево решений обучается на случайной подвыборке обучающего набора данных. 2) Для классификации нового изображения используется голосование деревьев решений. Изображение классифицируется в класс, который получает наибольшее количество голосов.

Преимущества Random Forest в задачах классификации изображений:

1. **Выбор признаков:** Random Forest автоматически выбирает случайные подмножества признаков при построении каждого дерева решений. Это способствует разнообразию моделей и уменьшает корреляцию между деревьями.

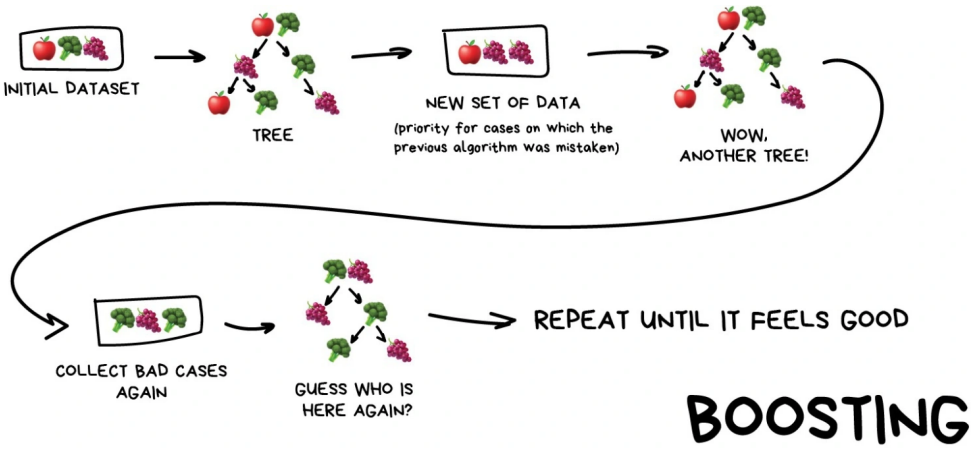
2. **Комбинирование решений:** Каждое дерево прогнозирует класс объекта, и затем Random Forest комбинирует эти прогнозы для получения окончательного решения. Обычно используется голосование большинства для определения класса.

3. **Устойчивость к переобучению:** Random Forest имеет встроенный механизм управления переобучением благодаря случайному выбору признаков и использованию ансамбля деревьев.

6.3.5 Ensemble

Определение 21. Ensemble - это более общий термин, который относится к любому методу, который объединяет результаты нескольких моделей. Существует множество различных методов ансамблирования, включая boosting, stacking и bagging.

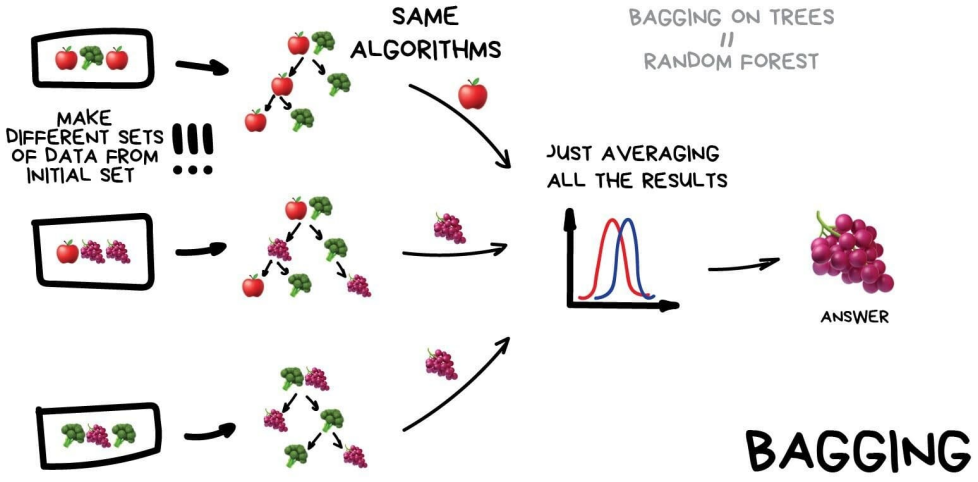
Определение 22. Boosting - это метод, в котором каждый новый классификатор обучается на ошибках предыдущих классификаторов. Это приводит к тому, что каждый новый классификатор фокусируется на ошибках предыдущих классификаторов, что может привести к повышению точности классификации. Например, AdaBoost и Gradient Boosting.



Как работает boosting?

1. Из обучающего набора данных генерируется модель обучения.
2. Модель обучения используется для классификации изображений в обучающем наборе данных.
3. Изображения, которые были неправильно классифицированы моделью обучения, получают больший вес в следующем шаге.
4. Шаг 2-3 повторяются до тех пор, пока не будет достигнута желаемая точность.

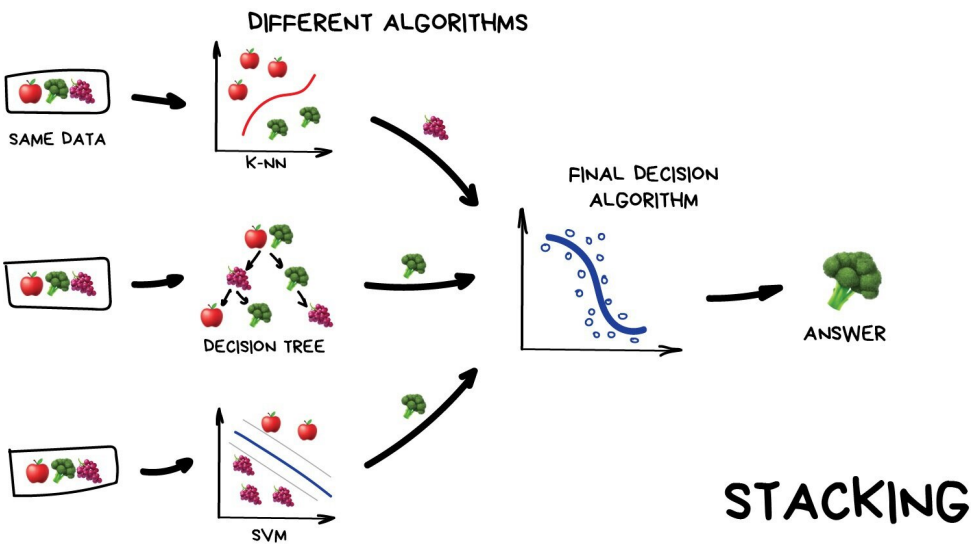
Определение 23. Bagging - это метод ансамблирования, который объединяет результаты нескольких моделей, обучаемых на различных подвыборках обучающего набора данных.



Как работает bagging?

1. Из обучающего набора данных генерируется несколько моделей обучения.
2. Для классификации нового изображения используются результаты нескольких моделей обучения.

Определение 24. Stacking - это метод ансамблирования, который объединяет результаты нескольких моделей, используя другую модель для прогнозирования окончательного результата.



Особенности применения boosting и ансамблинга в задачах распознавания изображений

Чтобы получить хорошие результаты с любой моделью, существуют определенные критерии (данные, гиперпараметры), которым необходимо соответствовать. Но в реальном сценарии вы можете либо получить плохие данные для обучения, либо столкнуться с трудностями при определении подходящих гиперпараметров. В таких ситуациях объединение нескольких моделей с более низкой производительностью может помочь получить нужные вам результаты. В некотором смысле, обучение в ансамбле можно рассматривать как способ компенсировать плохие алгоритмы обучения за счет выполнения большого количества дополнительных вычислений. С другой стороны, альтернативой является гораздо большее обучение на одной системе, не являющейся ансамблем. Комплексная система может быть более эффективной в повышении общей точности при одинаковом увеличении вычислительных ресурсов, ресурсов хранения или ресурсов связи за счет использования этого увеличения для двух или более методов, чем это было бы достигнуто за счет увеличения использования ресурсов для одного метода. Таким образом, они могут быть особенно полезны в следующих случаях:

- Когда набор данных для обучения является небольшим или несбалансированным.

- Когда необходимо распознавать объекты, которые могут быть повернуты, искажены или изменены другими способами.
- Когда необходимо распознавать объекты, которые имеют схожие признаки.

Темы для докладов

Для повышения оценки успешно сданного экзамена на балл можно сделать небольшой доклад о конкретном инструменте (плюс нужно сделать краткий мануал в конспект). Длительность доклада - 10-15 минут. Доклад состоит из основных возможностей инструмента, примера применения, небольшого сравнения с аналогами. Инструментами являются библиотеки, движки и даже конкретные ML-модели.

Рис. 17: Не, ну серьёзно.

Желающие – отписываетесь в Tg о том, какую тему взяли. И после записывайтесь в список в формате ФИО.Тема.Дата (белый шрифт у несогласованных тем):

0.

1.

2. Выродов Михаил. Нейросеть U-Net для семантической сегментации изображения. 05.12.23;

3. Задорожный Александр. Stable Diffusion Model. 12.12.23;

4. Старцев Матвей. Семейство RPG Maker. 05.12.23;

Лабораторная работа 1: Лабиринт

В первой лабораторной нужно эффективно решить онлайн-задачу определения возможности выхода из лабиринта.

Рис. 18: Классику нужно знать!

Лабиринт представляет собой набор перегородок расположенный на прямоугольной карте. Размер карты не менее $10^6 \times 10^6$. Для некоторых ячеек существует выход наружу, для остальных – нет. Необходимо научится быстро отвечать на запрос, можно ли выйти из лабиринта, если мы находимся в ячейке (x, y).

Асимптотические требования (N – Количество ячеек на карте):

- Время предподсчета: $O(NlogN)$.
- Дополнительная память: $O(N)$.
- Обработка запроса: $O(logN)$.

Требования к коду:

1. **Codestyle!**. Плохой код буду неанонимно Я.А. и К.К. отдавать.
2. **Тесты!**. Покрытие тестами не нужно большое, но нужно несколько содержательных тестов, подходящих под условие задачи.
3. Использование сторонних библиотек допустимо для отрисовки картинок (если хочется).
4. Ограничений по языкам – нет.

   .
5. Решение залить на git в репозиторий посвященный курсу в папку lab1.

В работе будут призы зрительских симпатий в виде дополнительных баллов за очень удачные решения.

Лабораторная работа 2: Камнем по голове.

Вторая лабораторная заключается в создании максимально реалистичного видеоролика, содержащего процесс кидания камня в стену.

Подробно описать процесс создания ролика. Какими инструментами и как пользовались. В чём преимущества вашего решения

Комментарии к ролику:

- Реализм понимается в терминах кинематографичности и/или физичности.
- Камень можно считать сферичным, но лучше не стоит.
- Стену можно считать геометрически плоской. Но визуально добавить объем текстурой.

Рис. 19: Как-то так...

В работе будут призы зрительских симпатий в виде дополнительных баллов за очень удачные решения.

Лабораторная работа 3: Todo: Firefox.

Третья лабораторная по приятнее. Реализовать систему компьютерного зрения, распознающую малых панд. Добиться не менее 65% верно-положительных и верно-отрицательных результатов.

Кроме непосредственно кода с реализациями/применением/обучением моделей компьютерного зрения сделать отчёт.

Комментарии к ролику:

- Описанием моделей и обоснованием их выбора
- Описать использованные для обучения (если было) и валидации(а вот это обязательно) моделей.
- Обосновать достижение требуемой точности.

Рис. 20: Хотелось бы...

В работе будут призы зрительских симпатий в виде дополнительных баллов за очень удачные решения.

Список литературы

[1] Franco Preparata and Michael Shamos. *Computational Geometry*. 01 1985.