

Нахождение контуров и операции с ними

Как мы можем получить контуры ?

- Детекторы границ
- Биноризация изображения
- Морфологические операции (эрозия, диллатация)

Нахождение контуров и операции с ними

Контурный анализ — это метод описания, хранения, распознавания, сравнения и поиска графических объектов по их контурам.



Ограничения на область применения контурного анализа:

- ✓ объект может не иметь чёткой границы, или может быть зашумлён помехами, что приводит к невозможности выделения контура;
- ✓ перекрытие объектов или их группировка приводит к тому, что контур выделяется неправильно и не соответствует границе объекта.



Переход к рассмотрению только контуров объектов позволяет уйти от пространства изображения – к пространству контуров, что **существенно снижает сложность алгоритмов и вычислений**.

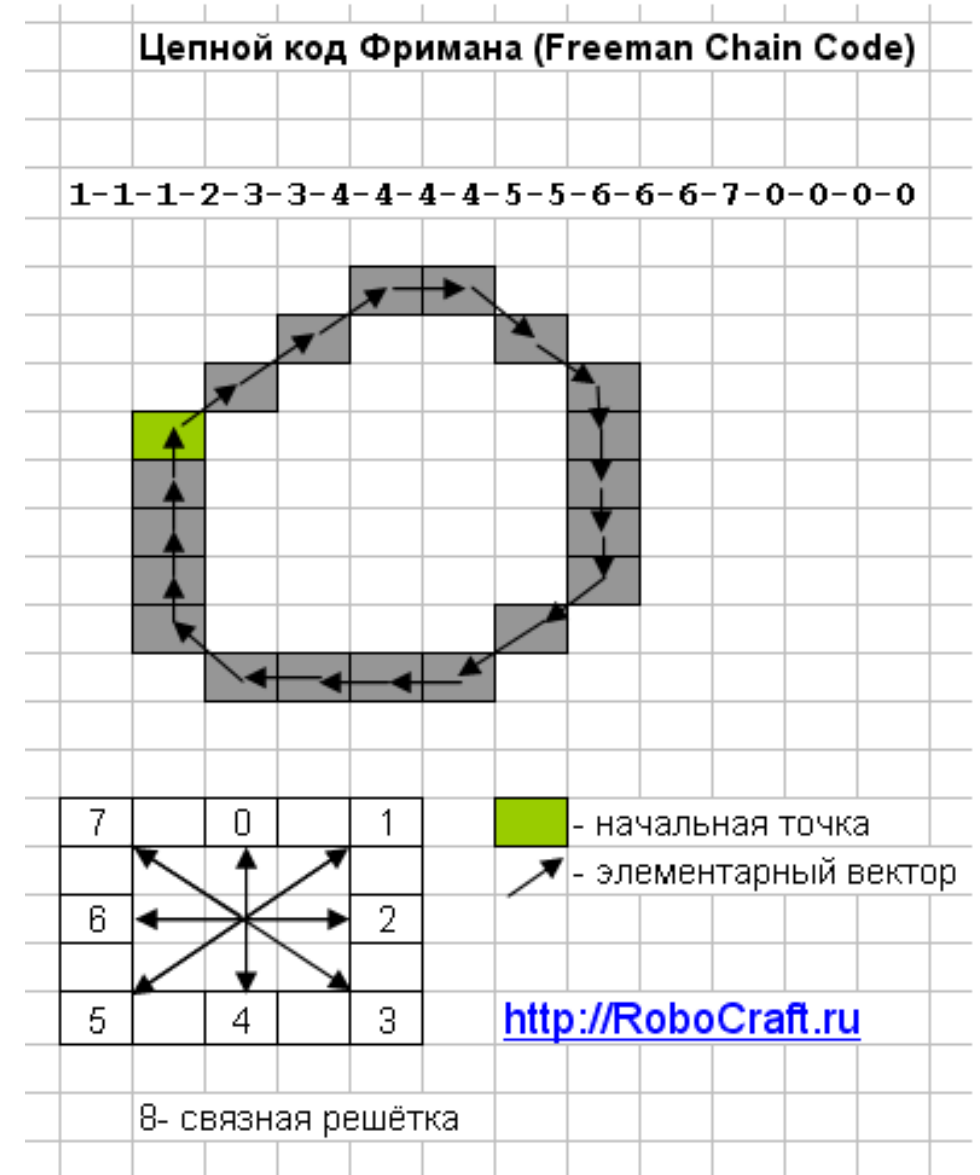
Вывод:

- контурный анализ имеет довольно **слабую устойчивость к помехам**;
- простотой и быстрый;
- хорошо работает при чётко выраженном объекте на контрастном фоне и отсутствии помех.

Нахождение контуров и операции с ними

Цепной код Фримена (Фридмана) (Freeman Chain Code)

Цепные коды применяются для представления границы в виде последовательности отрезков прямых линий определённой длины и направления. В основе этого представления лежит 4- или 8- связная решётка. Длина каждого отрезка определяется разрешением решётки, а направления задаются выбранным кодом.

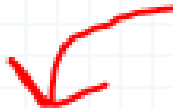


Нахождение контуров и операции с ними

Для поиска контуров используется функция **cvFindContours()**:

Функция **cvFindContours()** может находить **внешние и вложенные контуры** и **определять их иерархию вложения**.

```
CVAPI(int)  cvFindContours( CvArr* image, CvMemStorage* storage, CvSeq** first_contour,
                           int header_size CV_DEFAULT(sizeof(CvContour)),
                           int mode CV_DEFAULT(CV_RETR_LIST),
                           int method CV_DEFAULT(CV_CHAIN_APPROX_SIMPLE),
                           CvPoint offset CV_DEFAULT(cvPoint(0,0)));
```



```
#define CV_RETR_EXTERNAL 0 // найти только крайние внешние контуры
#define CV_RETR_LIST     1 // найти все контуры и разместить их списком
#define CV_RETR_CCOMP    2 // найти все контуры и разместить их в виде 2-уровневой иерархии
#define CV_RETR_TREE     3 // найти все контуры и разместить их в иерархии вложенных контуров
```

Нахождение контуров и операции с ними

Последовательность действий при распознавании объектов методом контурного анализа:

1. предварительная обработка изображения (сглаживание, фильтрация помех, увеличение контраста);
2. бинаризация изображения;
3. операции математической морфологии;
4. выделение контуров объектов;
5. первичная фильтрация контуров (по периметру, площади и т.п.);
6. эквализация контуров (приведение к единой длине, сглаживание) — позволяет добиться инвариантности к масштабу;
7. перебор всех найденных контуров и поиск шаблона, максимально похожего на данный контур (или же сортировка контуров по какому-либо признаку, например, площади).

Свойства контуров

```
CVAPI(double) cvContourArea( const CvArr* contour,  
                             CvSlice slice CV_DEFAULT(CV_WHOLE_SEQ));
```

— возвращает площадь контура

contour — контур (последовательность или массив вершин)

slice — начальная и конечные точки контура (по-умолчанию весь контур)

```
CVAPI(double) cvArcLength( const void* curve,  
                           CvSlice slice CV_DEFAULT(CV_WHOLE_SEQ),  
                           int is_closed CV_DEFAULT(-1));  
  
#define cvContourPerimeter( contour ) cvArcLength( contour, CV_WHOLE_SEQ, 1 )
```

— возвращает периметр контура или длину кривой (части контура)

curve — последовательность или массив точек кривой (контур)

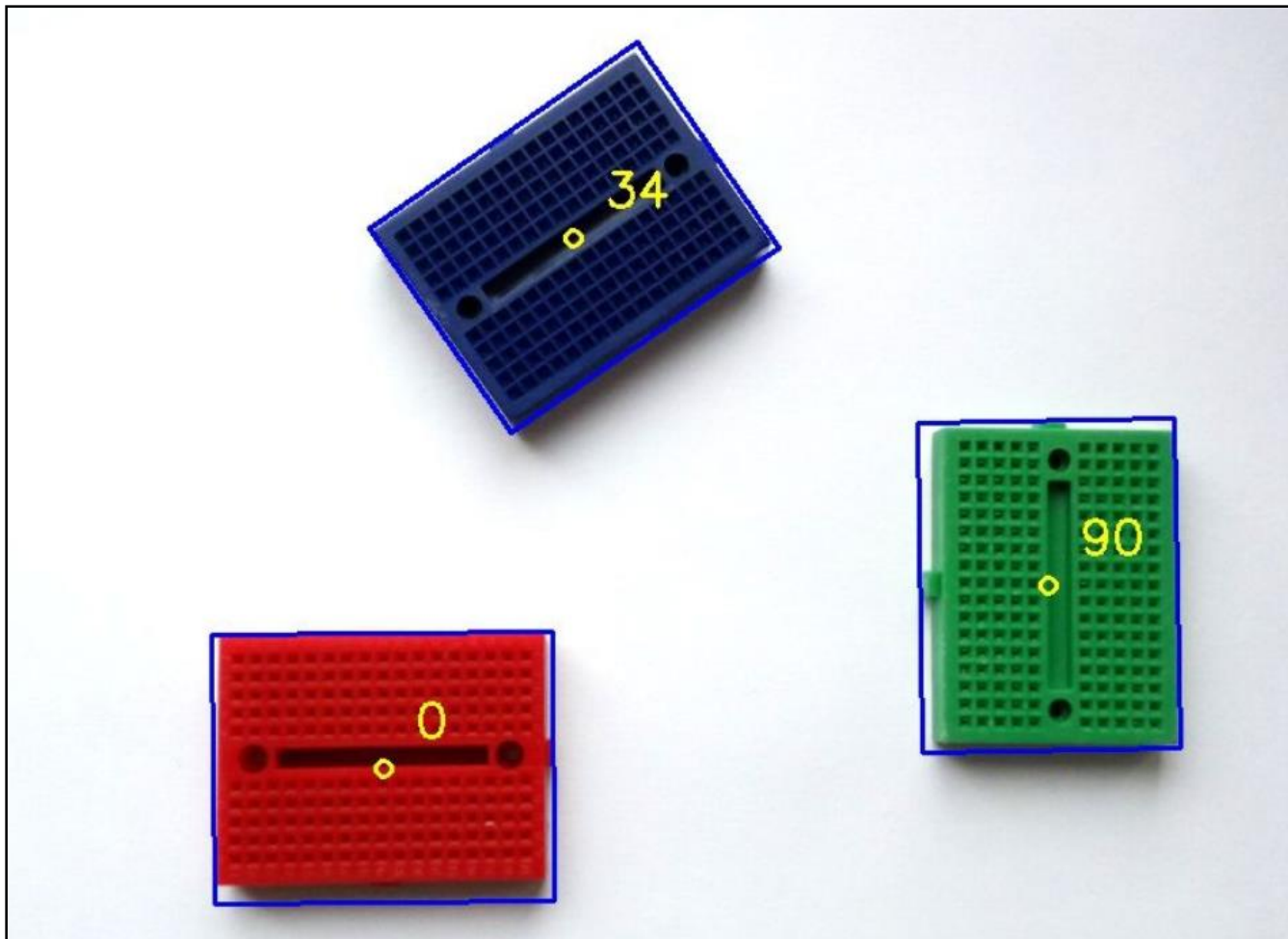
slice — начальная и конечные точки контура (по-умолчанию весь контур)

is_closed — определяет закрыта кривая или нет:

is_closed = 0 — кривая полагается открытой

is_closed > 0 — кривая полагается закрытой

пример применения контурного анализа для поиска прямоугольников
определения их наклона относительно горизонта (для роботов)



Преобразование Хафа

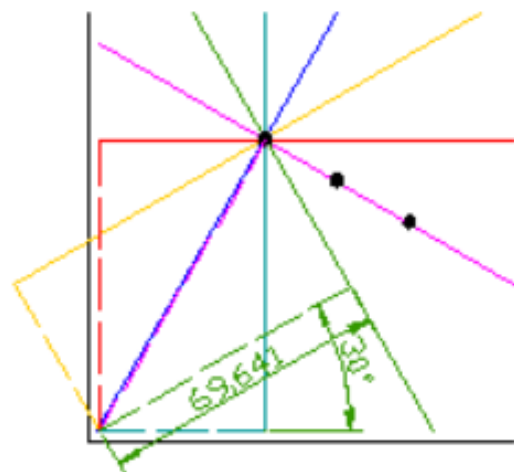
Преобразование Хафа (Hough Transform) — это метод для поиска линий, кругов и других простых форм на изображении.

Преобразование Хафа основывается на представлении искомого объекта в виде **параметрического уравнения**. Параметры этого уравнения представляют фазовое пространство (пространство Хафа).

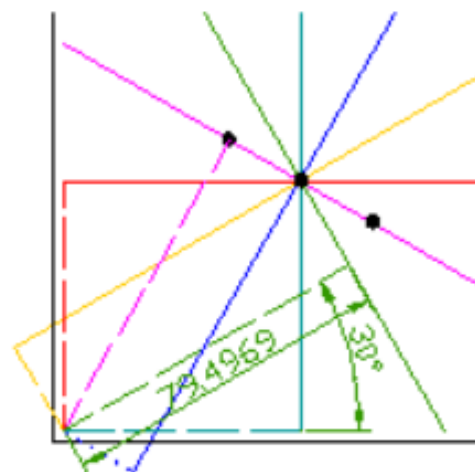
Берётся двоичное изображение (например, результат работы детектора границ Кенни). Перебираются все точки границ и делается предположение, что точка принадлежит линии искомого объекта — т.о. для каждой точки изображения рассчитывается нужное уравнение и получаются необходимые параметры, которые сохраняются в пространстве Хафа.

Финальным шагом является обход пространства Хафа и выбор максимальных значений, за которые «проголосовало» больше всего пикселей картинки, что и даёт нам параметры для уравнений искомого объекта.

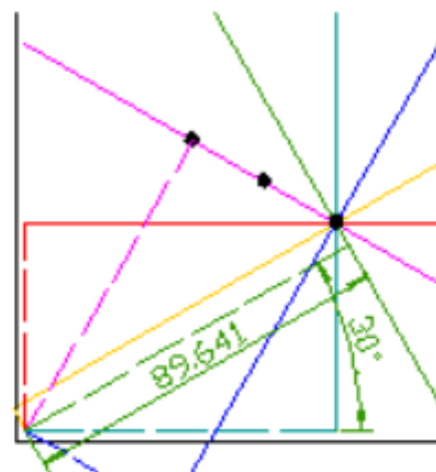
Рассмотрим исходное тестовое изображение из трех черных точек. Проверим, расположены ли точки на прямой линии.



Angle	Dist.
0	40
30	69.6
60	81.2
90	70
120	40.6
150	0.4



Angle	Dist.
0	57.1
30	79.5
60	80.5
90	60
120	23.4
150	-19.5



Angle	Dist.
0	74.6
30	89.6
60	80.6
90	50
120	6.0
150	-39.6

- Через каждую точку проведено (для наглядности) только по шесть прямых, имеющих разный угол.
- К каждой прямой из начала координат построен перпендикуляр.
- Для всех прямых длина соответствующего перпендикуляра и его угол с осью абсцисс сведены в таблицу.
- Данные таблицы являются результатом преобразования Хафа и могут служить основой для графического представления в "пространстве Хафа".

Прямую на плоскости можно представить:

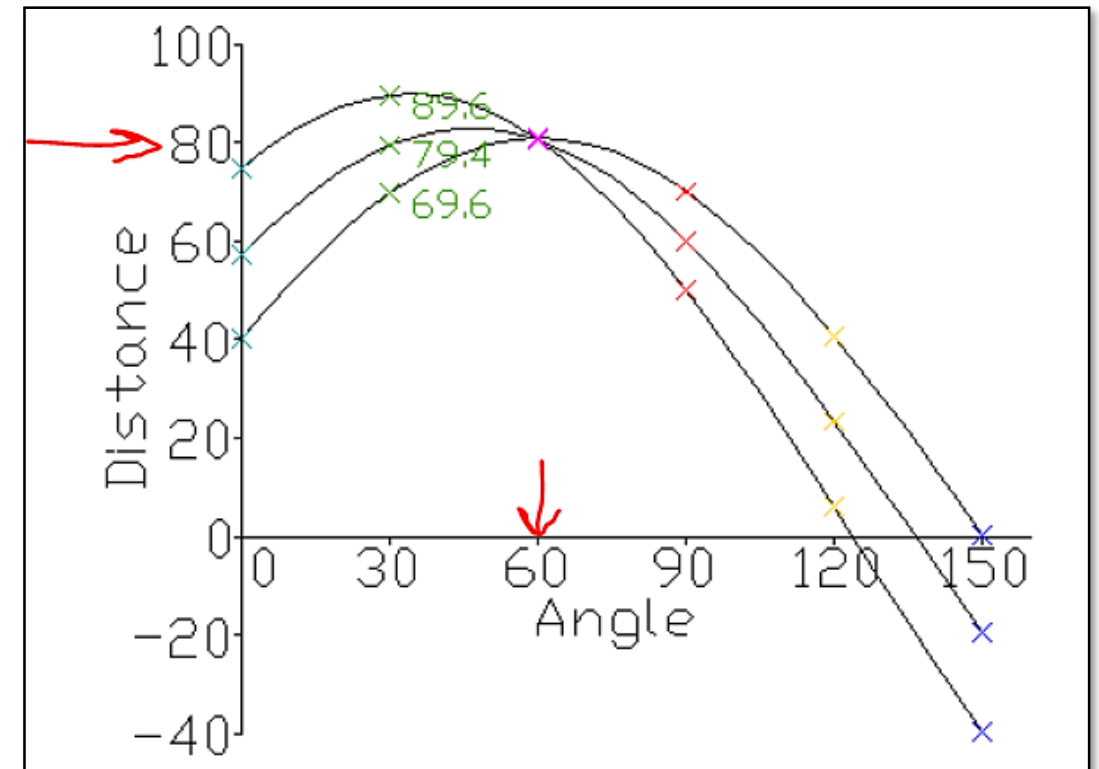
$$x \cdot \cos(f) + y \cdot \sin(f) = R$$

, где

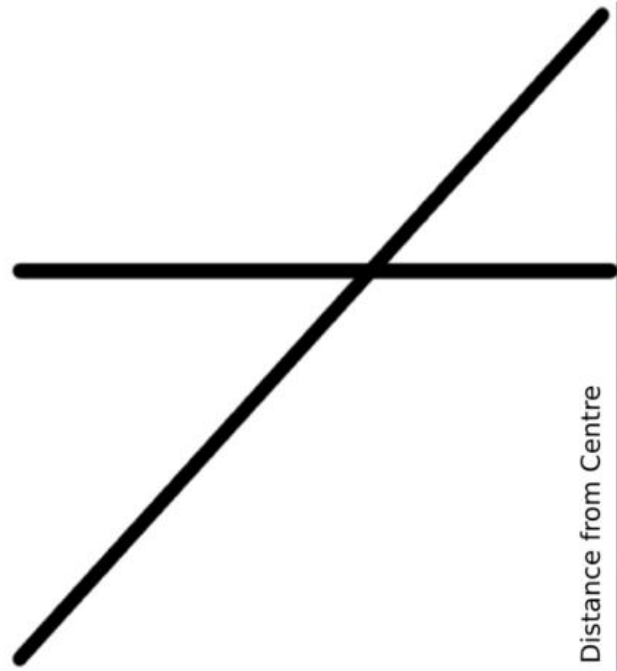
R — длина перпендикуляра опущенного на прямую из начала координат,
 f — угол между перпендикуляром к прямой и осью OX .

Через каждую точку (x, y) изображения можно провести несколько прямых с разными R и f , то есть каждой точке (x, y) изображения соответствует набор точек в фазовом пространстве (R, f) , образующий синусоиду.

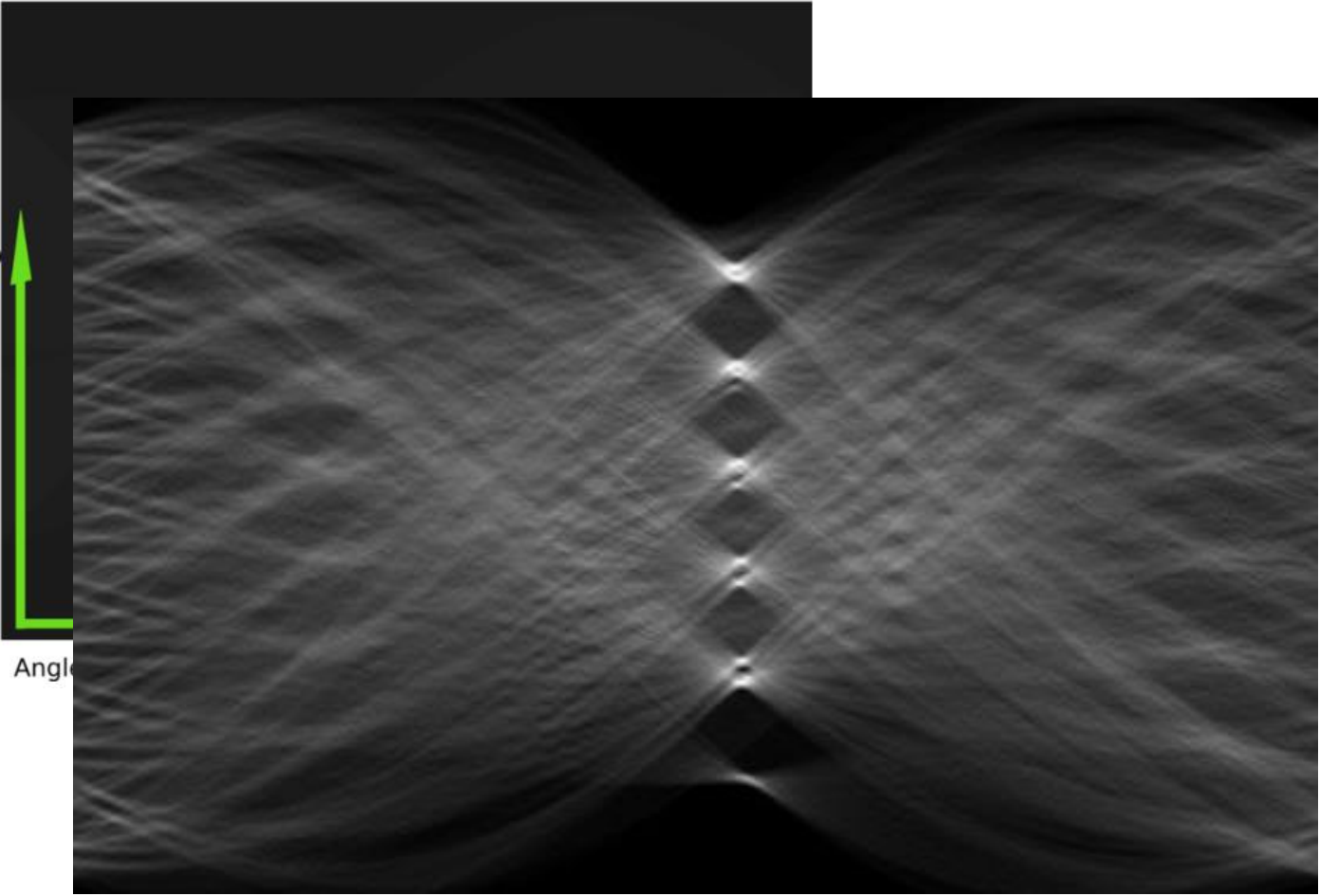
Координаты точки пересечения синусоид определяют параметры прямой, общей для проверяемых точек на исходном изображении.



Input Image



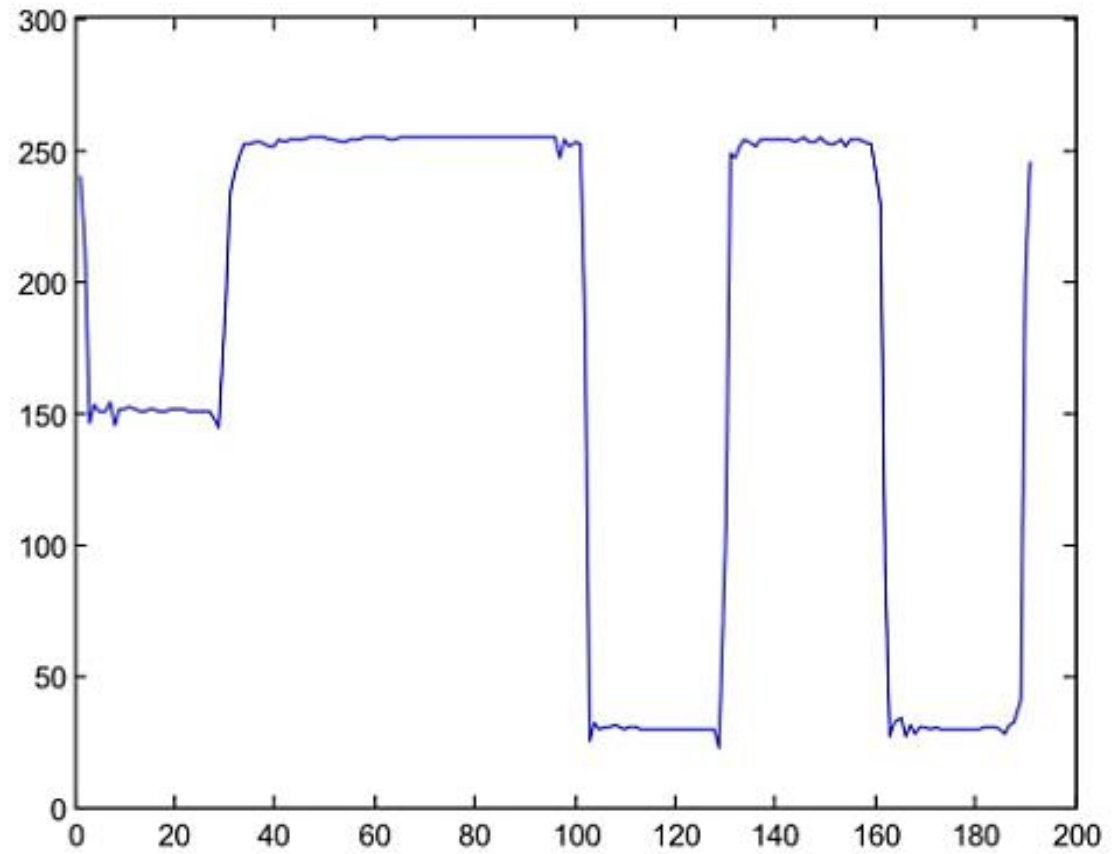
Rendering of Transform Results



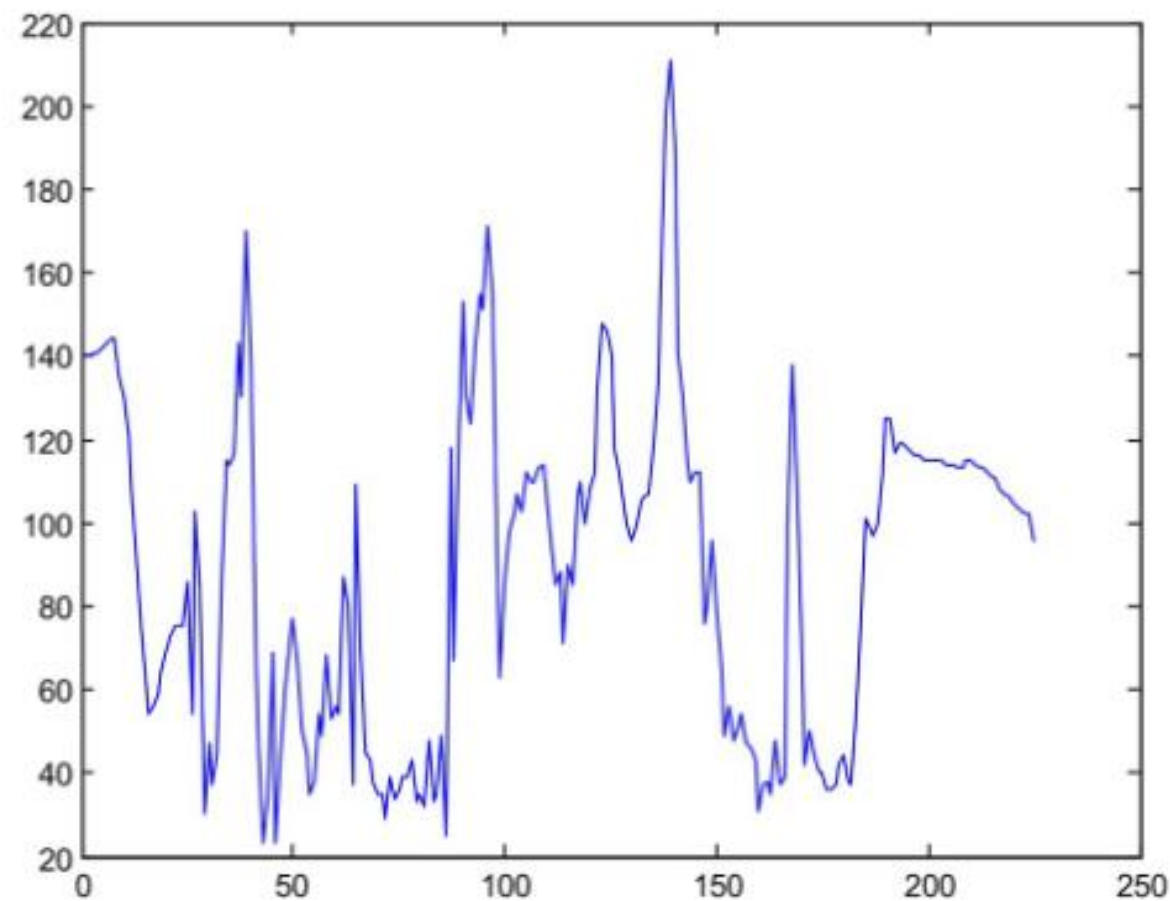
Distance from Centre

Angle

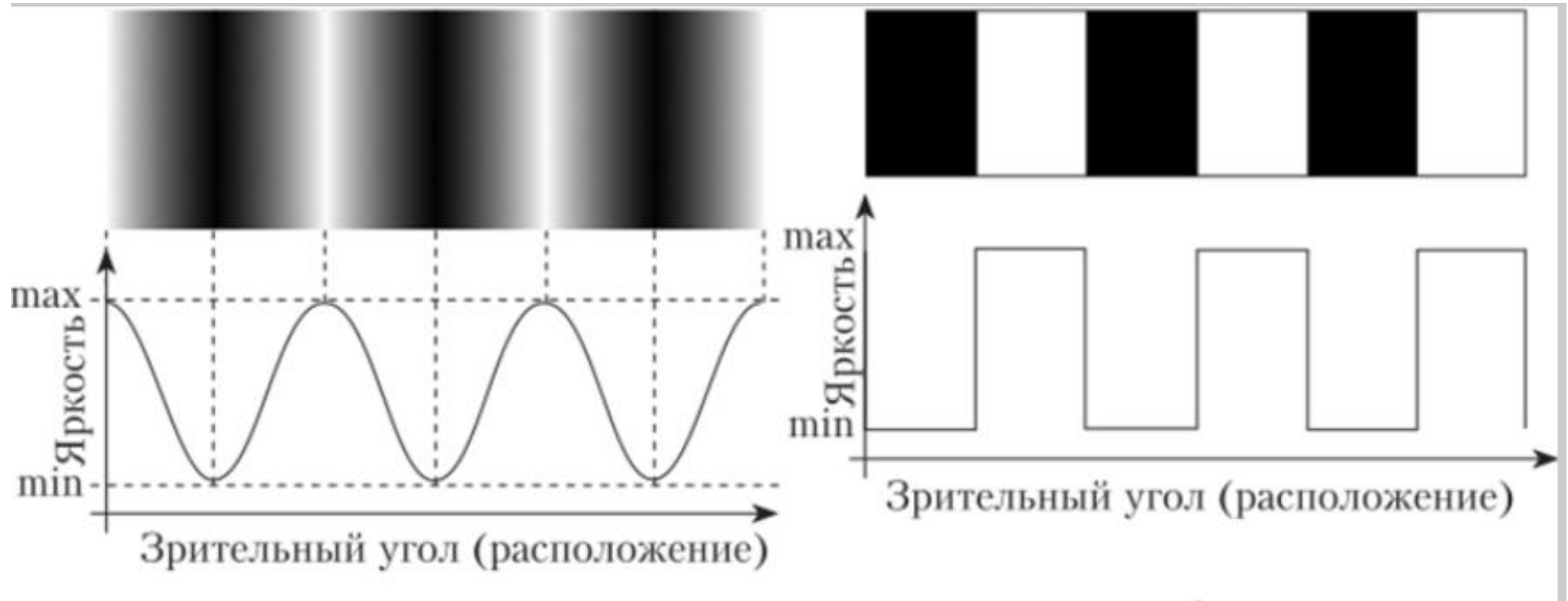
Частотное представление изображения



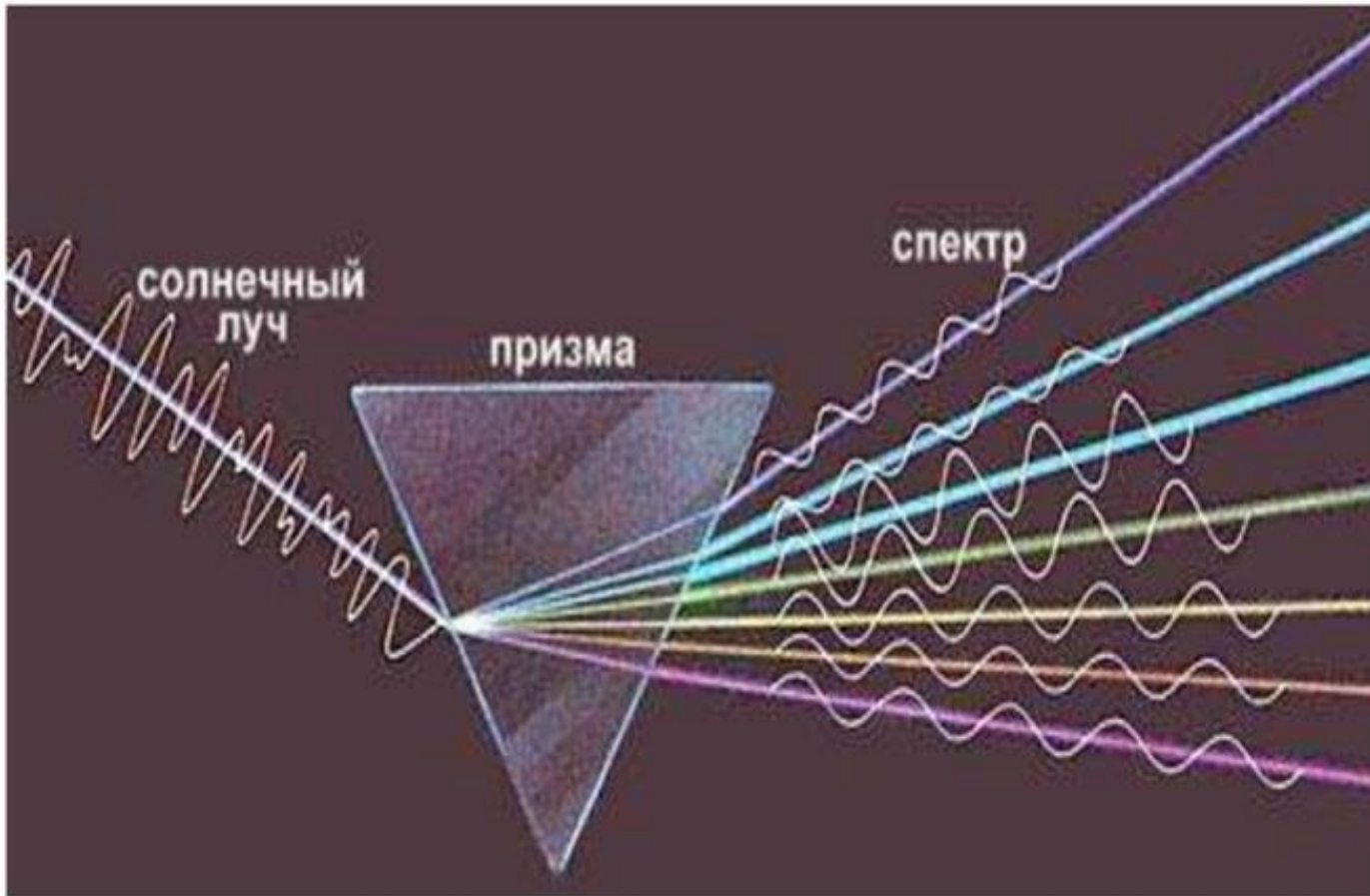
Частотное представление изображения



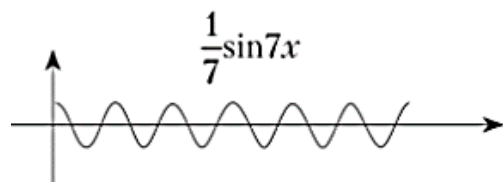
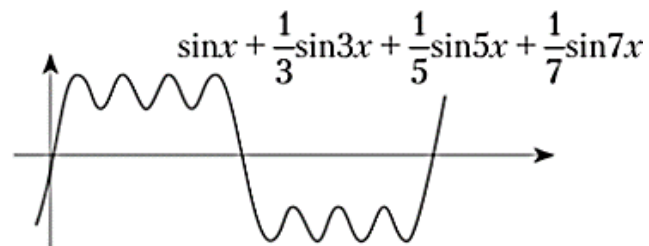
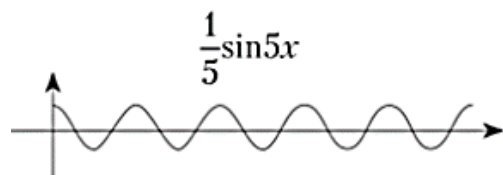
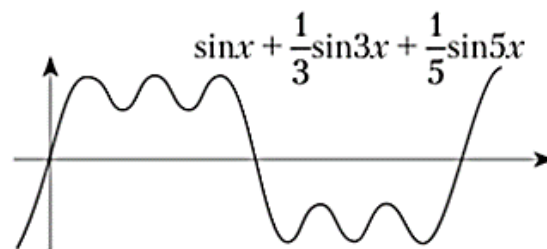
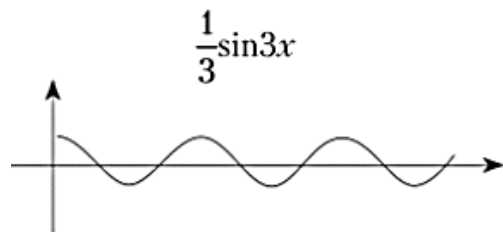
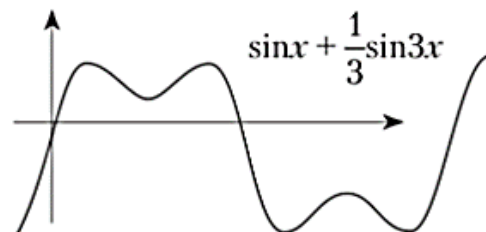
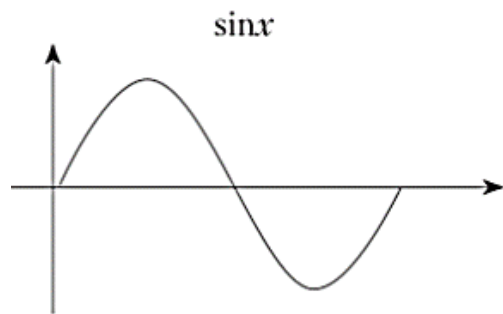
Частотное представление изображения



Представление изображения в виде ряда Фурье



Любое изображение м.б. представлено как сумма косинусов и синусов различной амплитуды и частоты.



Чем больше гармоник используется в разложении, тем точнее они воспроизводят импульс.

Но, чем меньше частота, тем меньший вклад она вносит, поэтому сумму можно принять конечной.

Двумерный случай

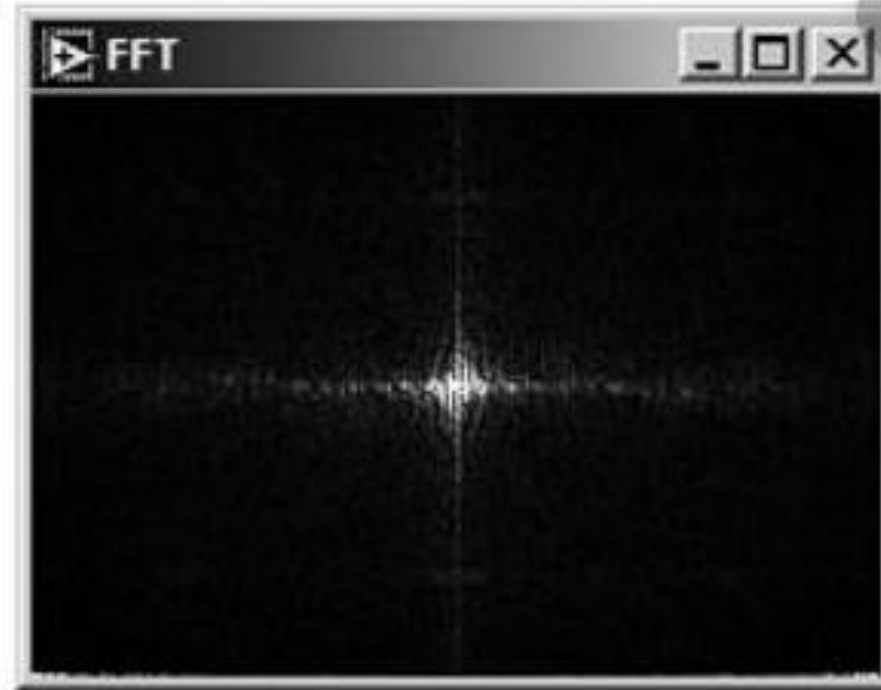
- Прямое Фурье-преобразование (Фурье-образ)

$$F(u, v) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x, y) e^{-i2\pi(ux+vy)} dx dy$$

- Обратное Фурье-преобразование

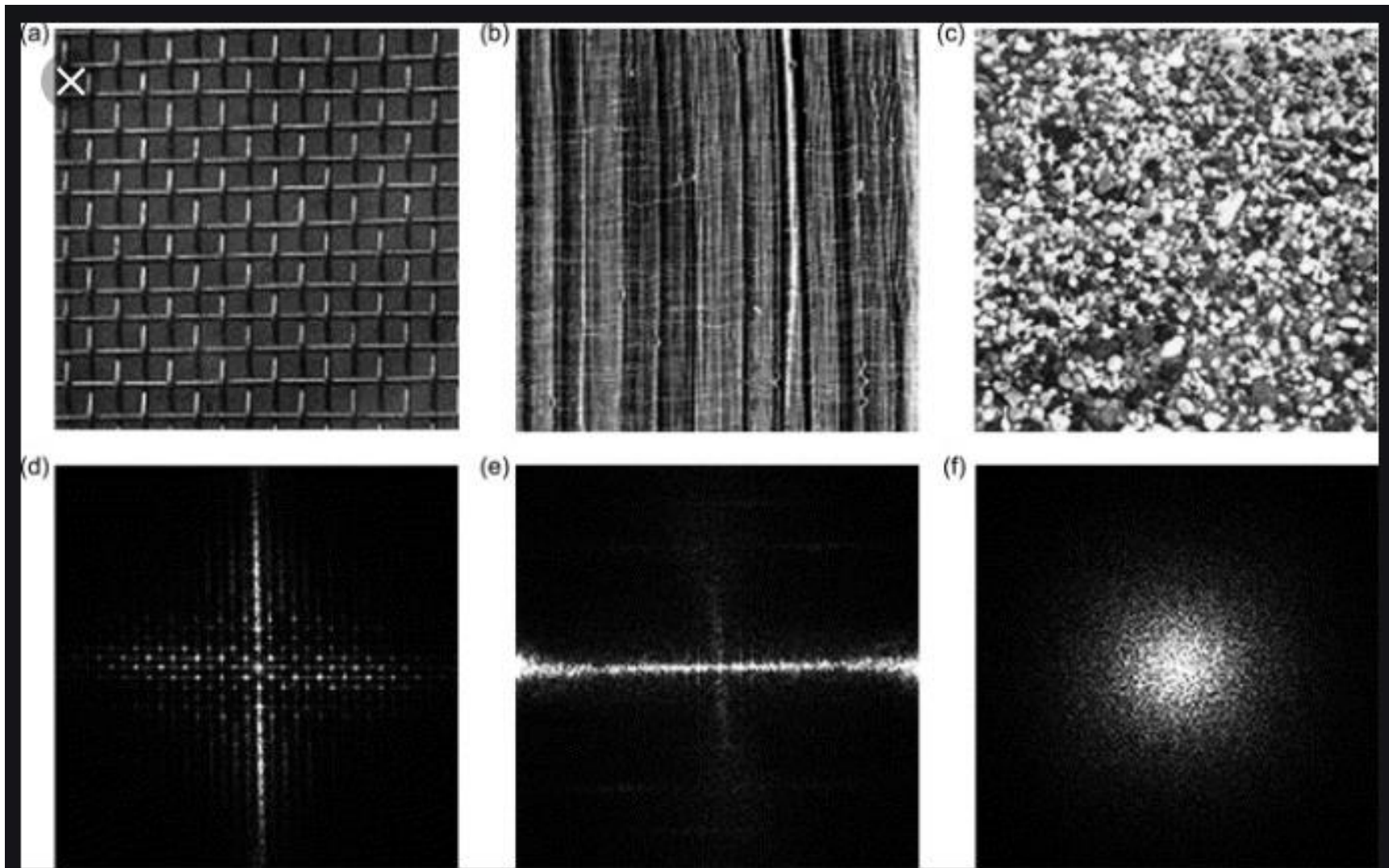
$$f(x, y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} F(u, v) e^{i2\pi(ux+vy)} du dv$$

Спектральный анализ изображений



- Низкие частоты (вблизи начала координат) соответствуют медленно меняющимся компонентам изображения
- Высокие – быстро меняющимся

Выявление текстур

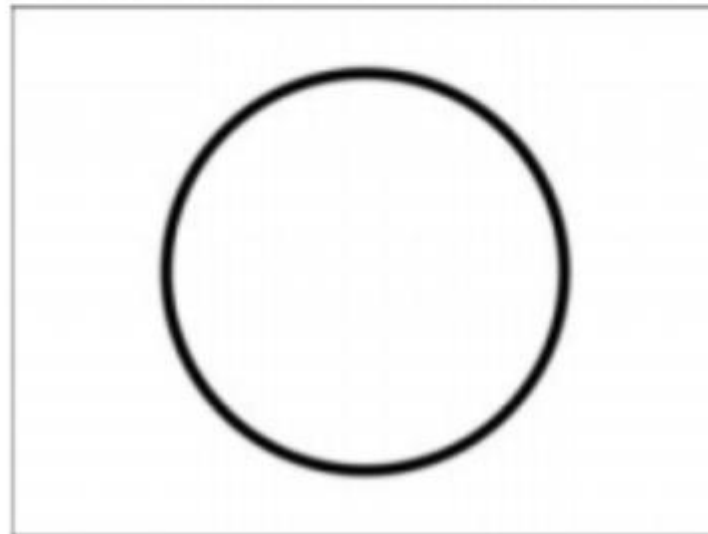
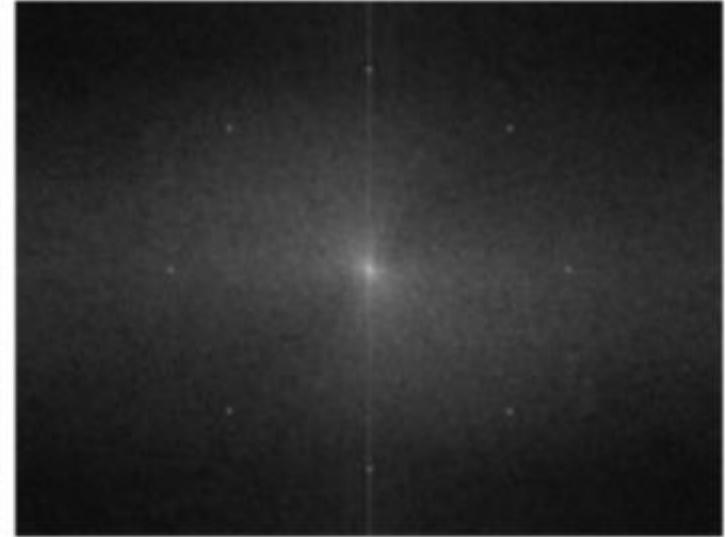


Поскольку тиснение очевидно дает пики в высоких частотах, их и видно в получившейся амплитуде. (Фото из архива NASA).

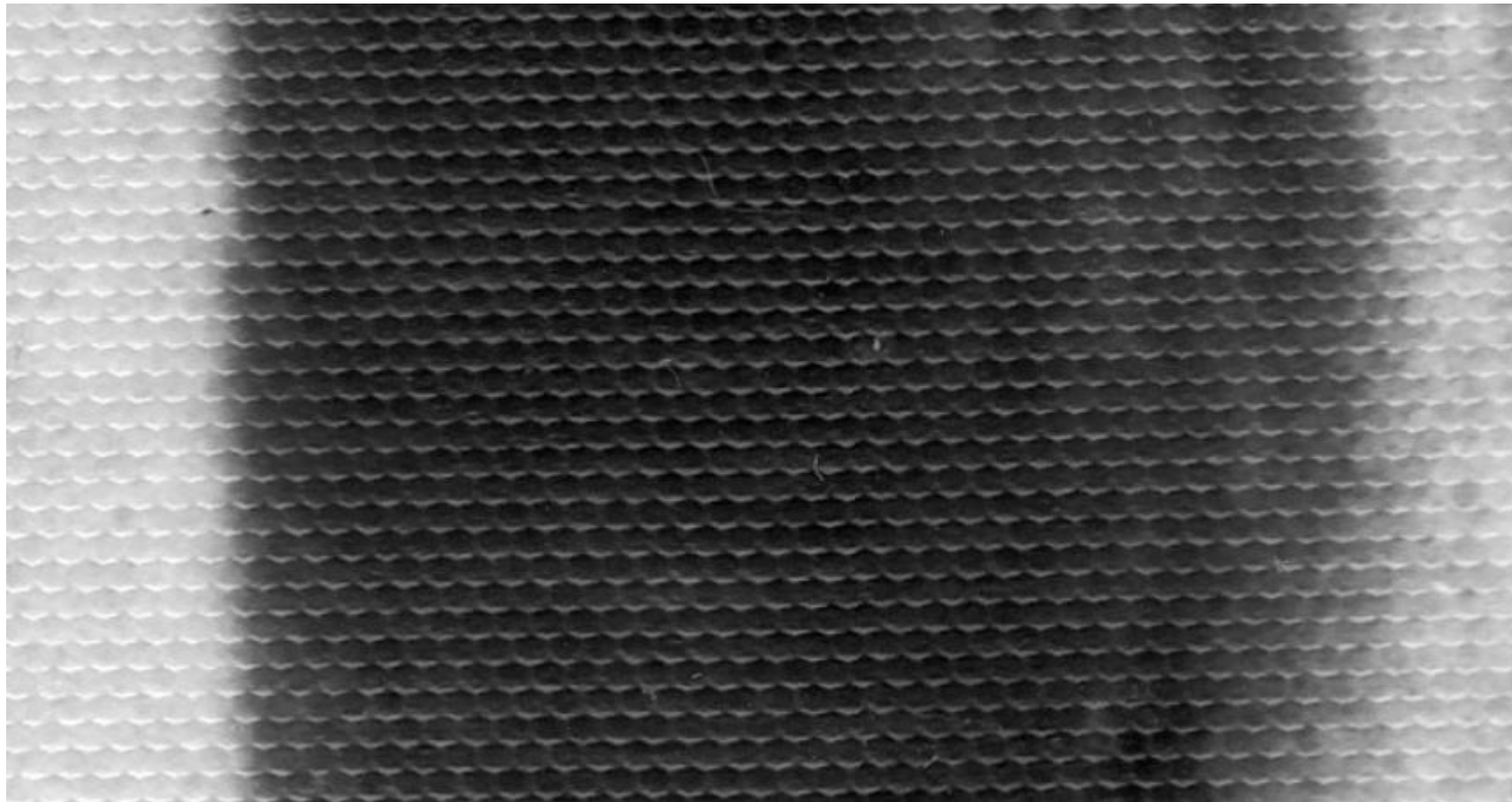
a b
c d

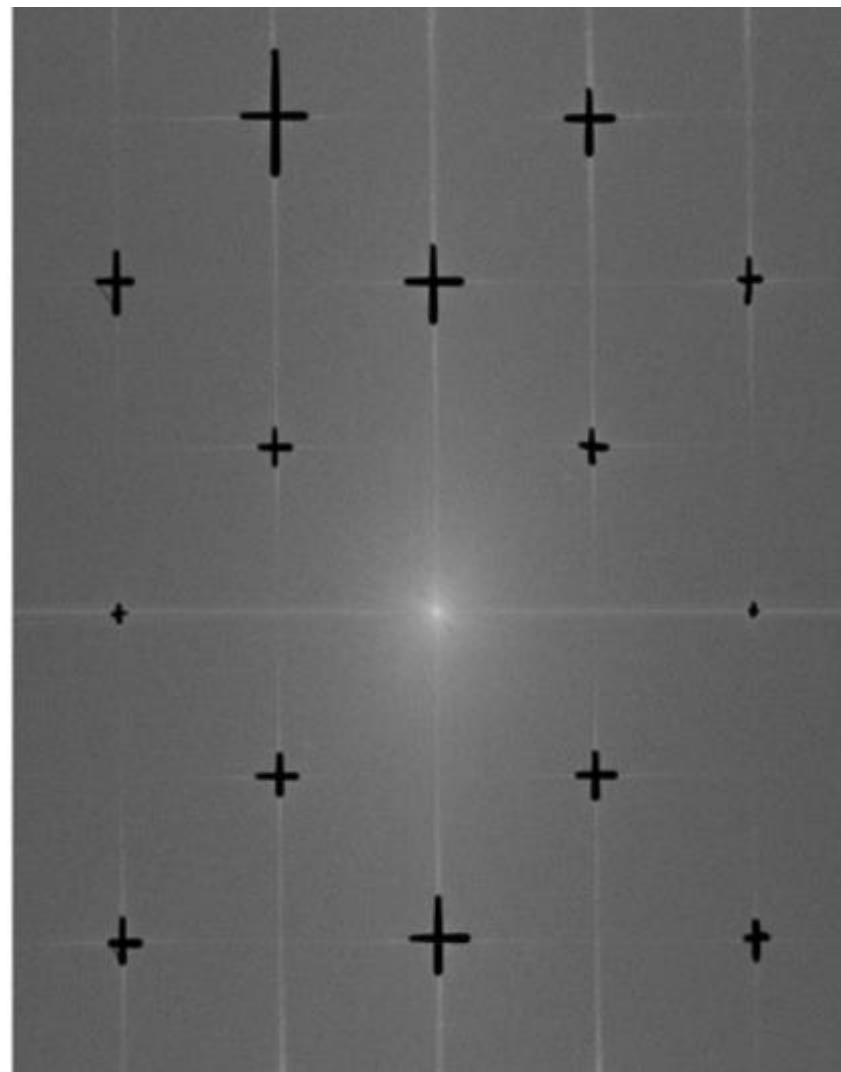
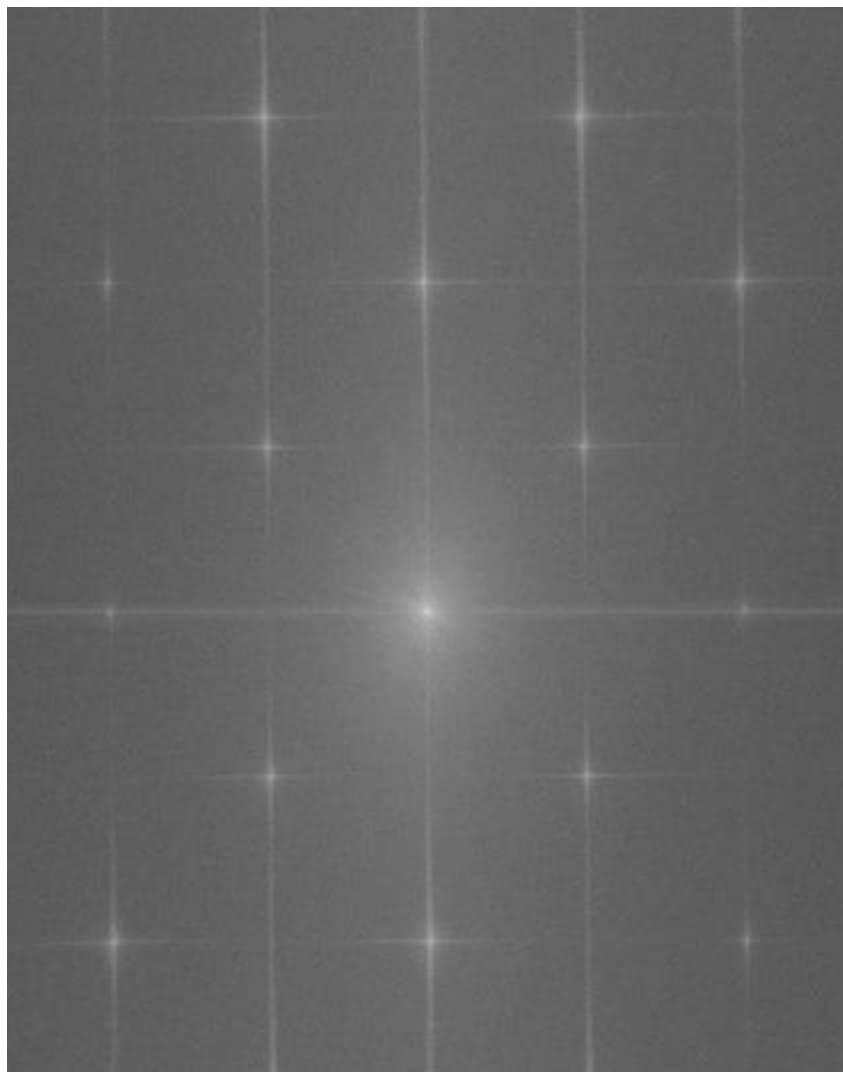
FIGURE 2.40

(a) Image corrupted by sinusoidal interference. (b) Magnitude of the Fourier transform showing the bursts of energy responsible for the interference. (c) Mask used to eliminate the energy bursts. (d) Result of computing the inverse of the modified Fourier transform. (Original image courtesy of NASA.)



Удаляем сетчатое тиснение с фото при помощи преобразований в ряд Фурье(плагина в фотошопе)

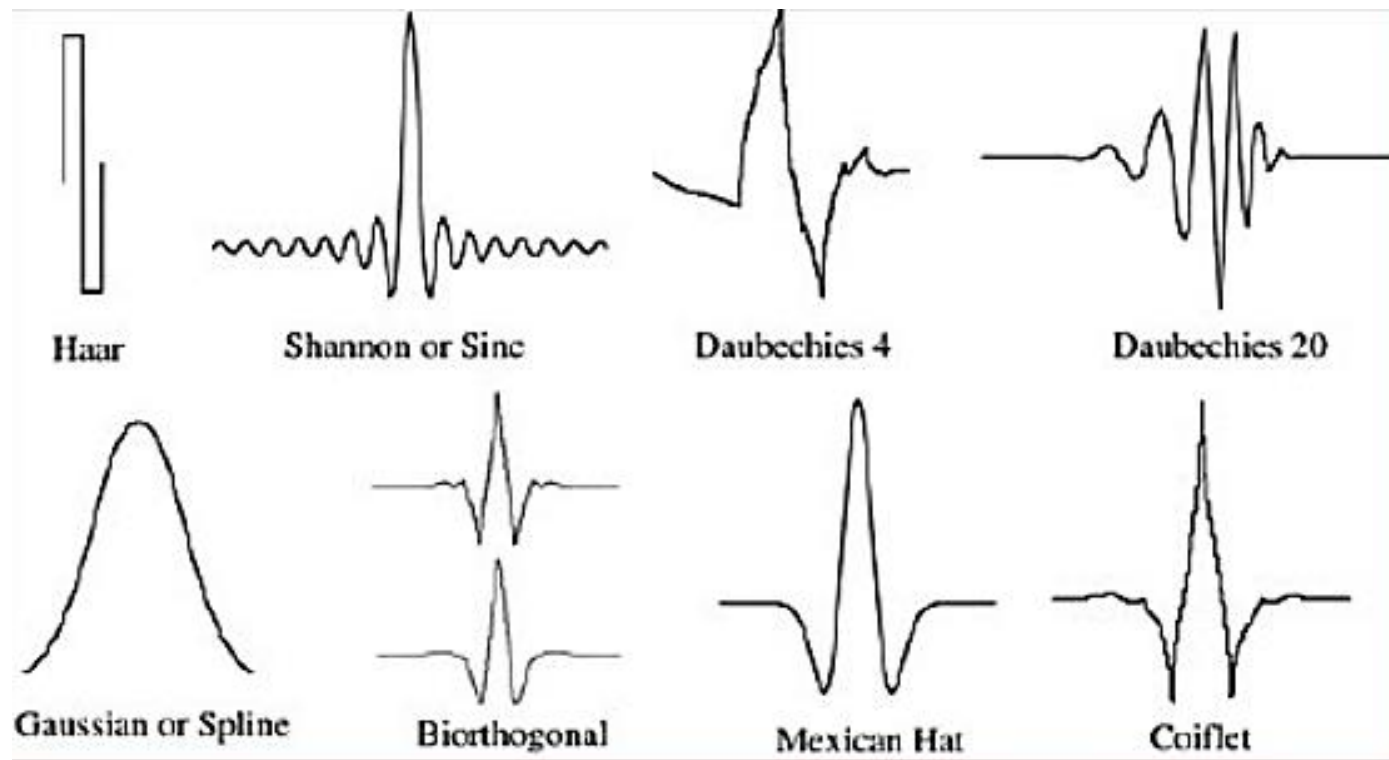




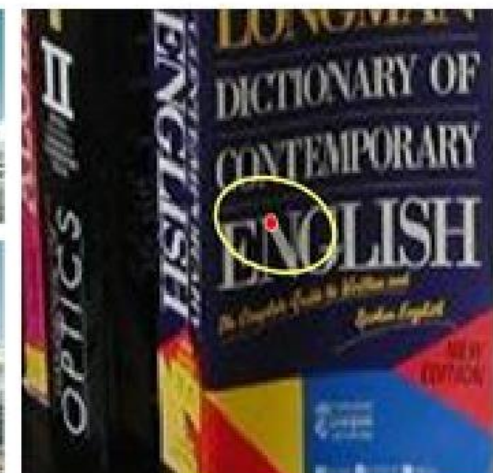
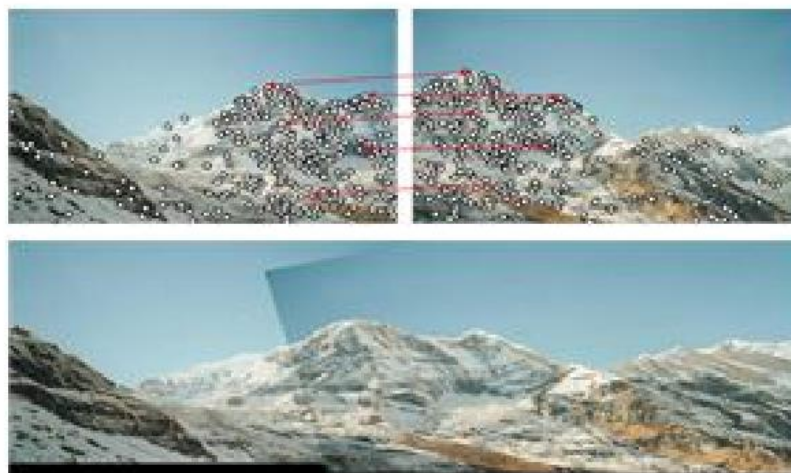
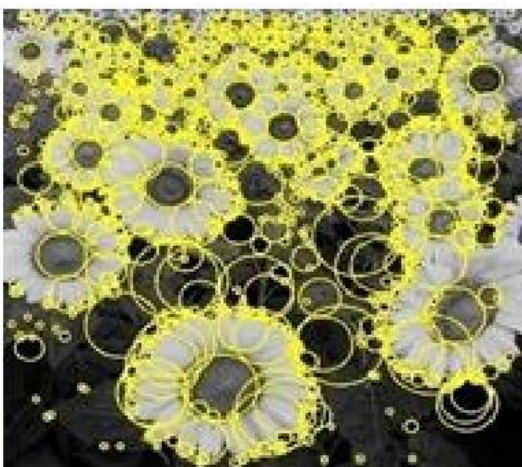
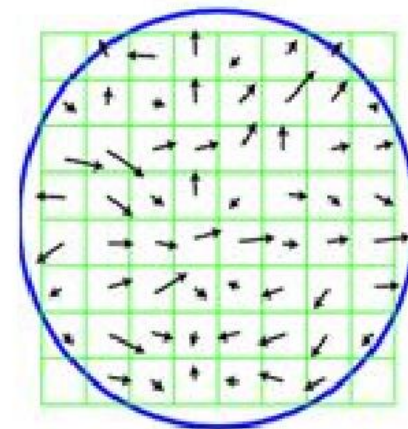
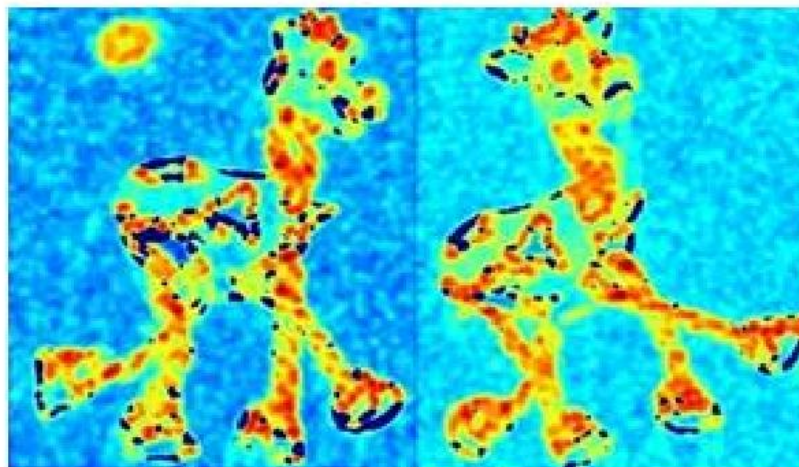
Применив обратное преобр Фурье получим



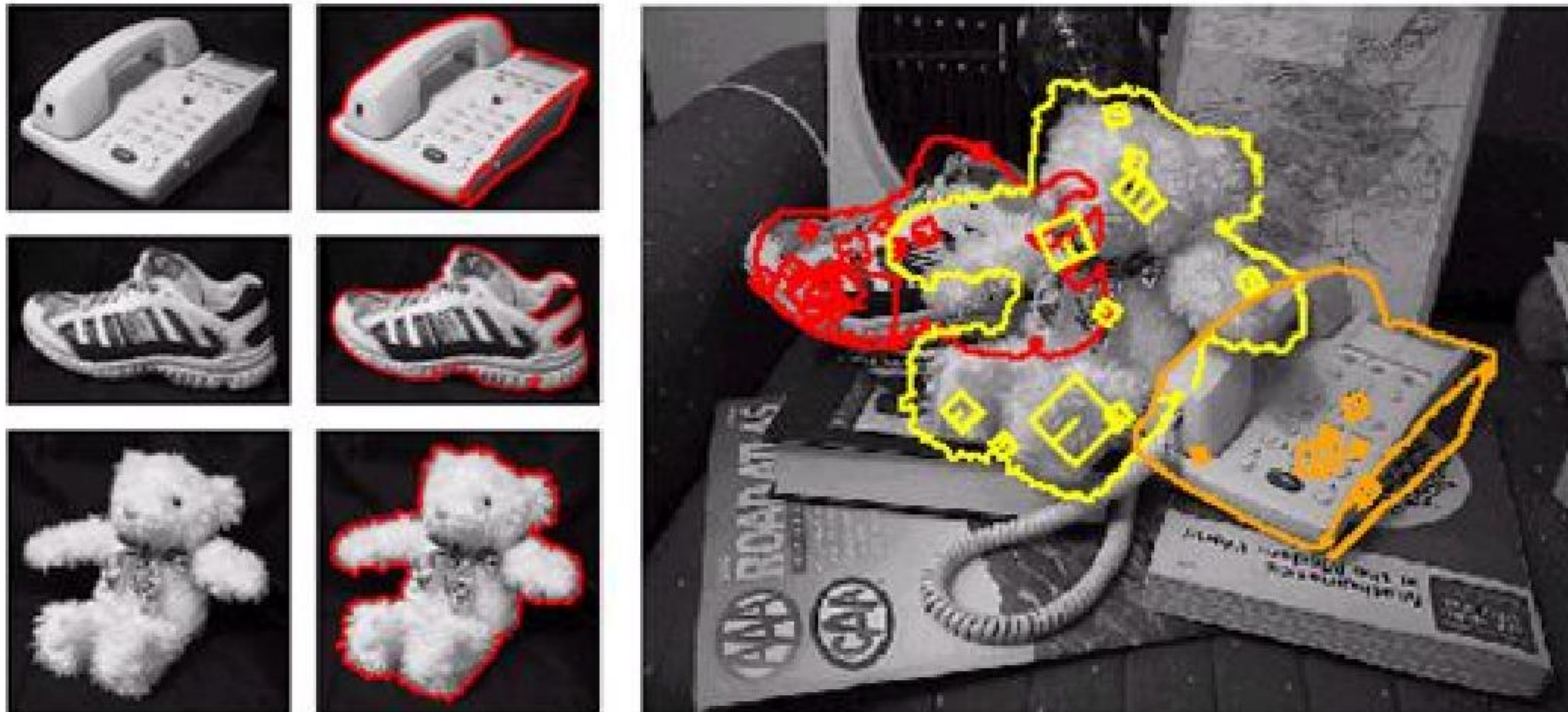
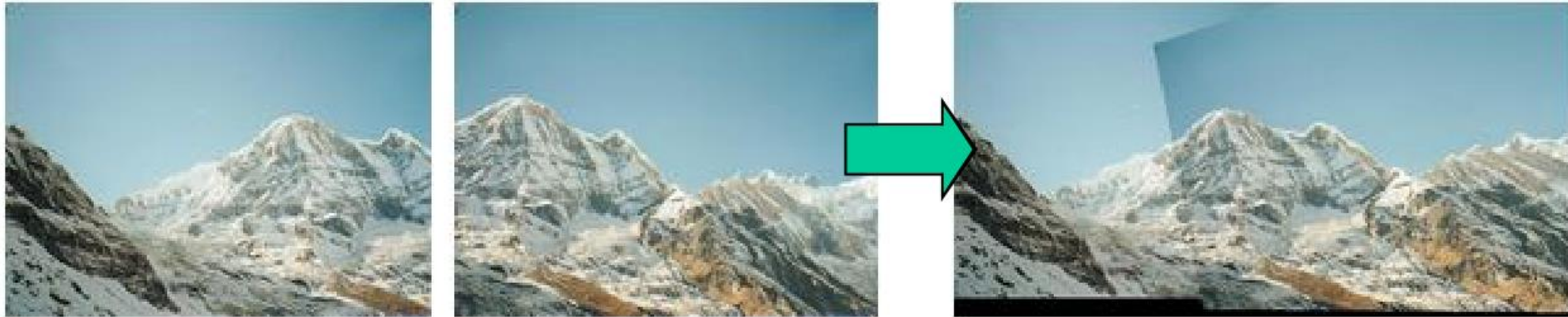
Вейвлеты



Локальные особенности

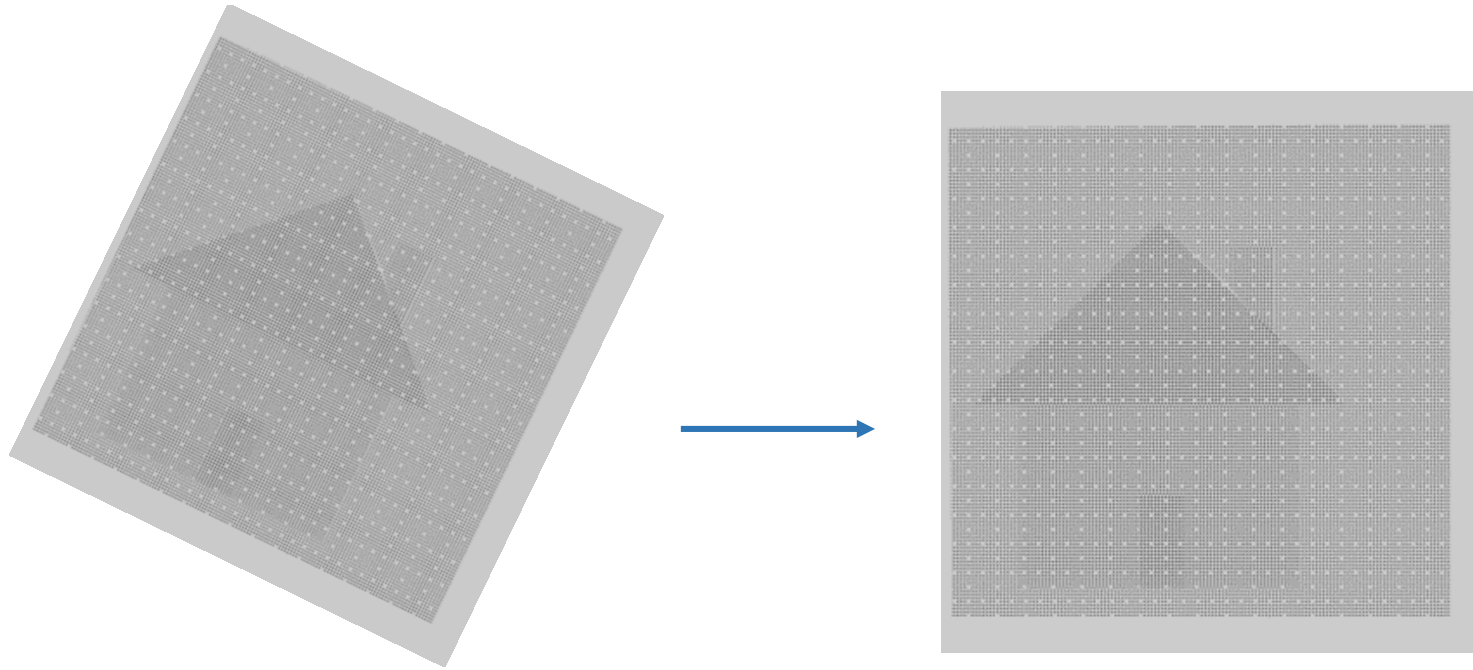


Задача сопоставления изображений



- Построение панорам
- Распознавание экземпляров объектов

Выравнивание изображений



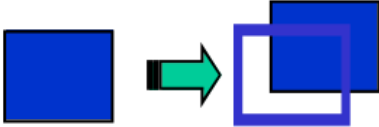

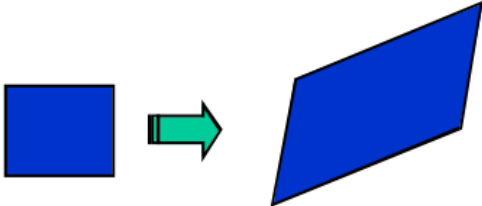
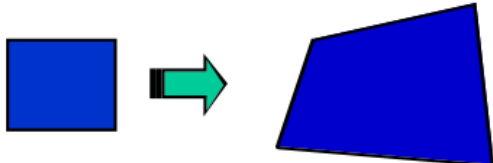
Есть два изображения
одного и того же
объекта. Как нам
совместить
изображения
автоматически?

Найдём такое преобразование (совмещение изображений), при котором изображения больше всего совпадут

Что нужно определить:

- Какое преобразование будем использовать?
- Как оценить совпадение (похожесть изображений)?

Геометрические преобразования

- Параллельный перенос 
- Подобие (перенос, масштаб, поворот) 
- Аффинное 
- Проективное (гомография) 

Функции сопоставления изображений

Попиксельное сравнение изображений:

$$\sum_X \sum_Y |I_1(X, Y) - I_2(X, Y)|$$

L1 метрика (SAD - Sum of absolute differences)

$$\sum_X \sum_Y (I_1(X, Y) - I_2(X, Y))^2$$

L2 метрика (SSD - Sum of squared differences)

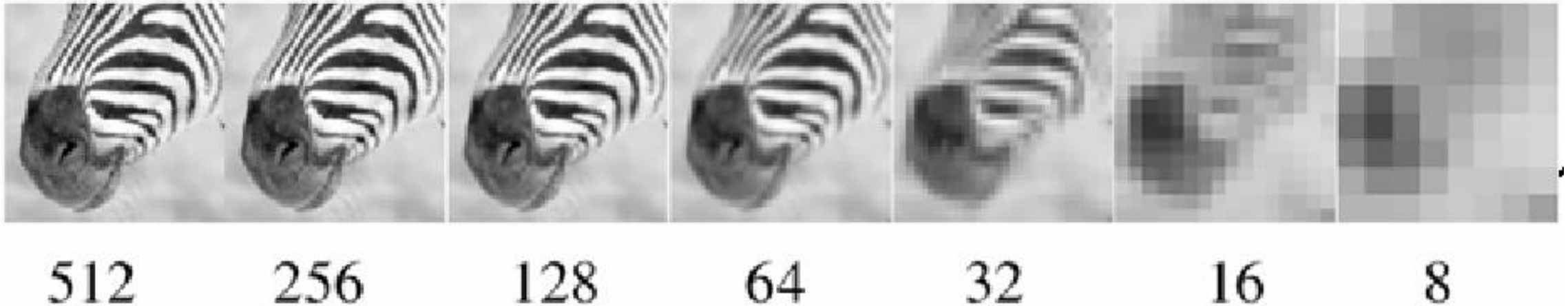
$$\sum_X \sum_Y I_1(X, Y) I_2(X, Y)$$

Кросс-корреляция (CC - Cross-correlation)

- SAD, SSD – минимизируются (0 – точное совпадение)
- CC – максимизируется (1 – точное совпадение)

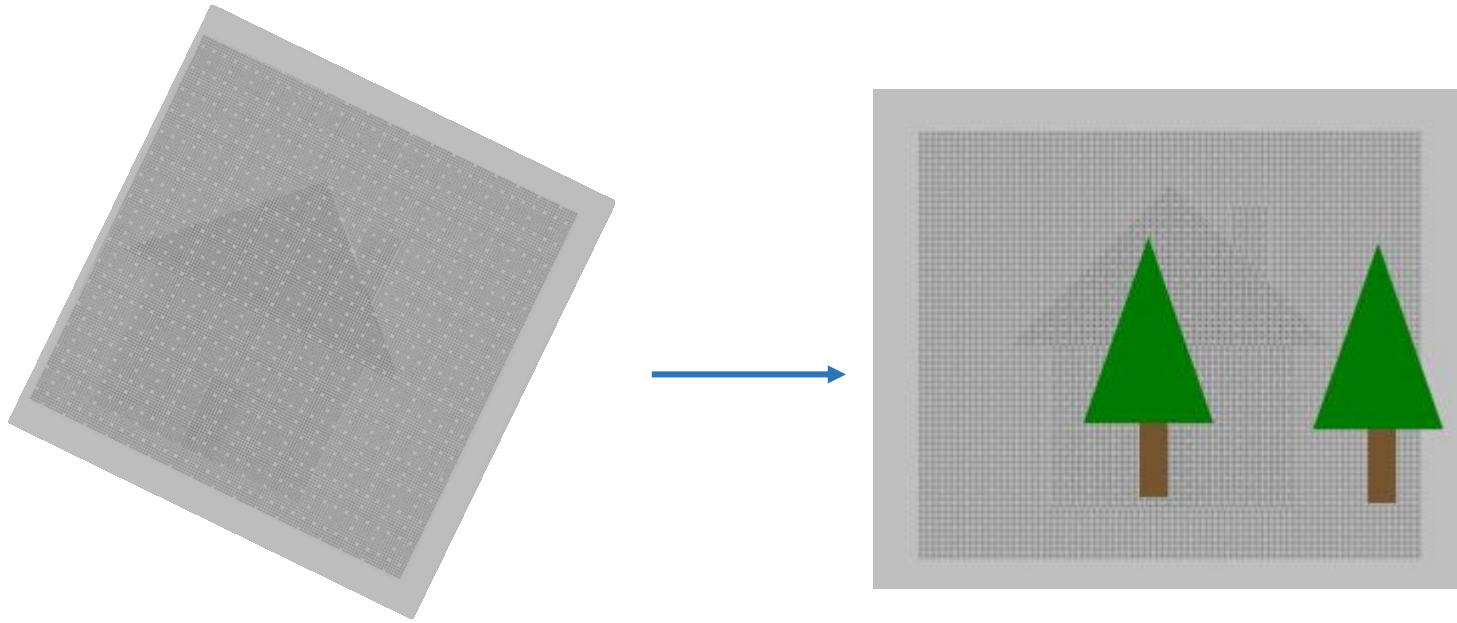
Многомасштабное сопоставление

Вариант полного перебора параметров можно улучшить с помощью многомасштабного сопоставления



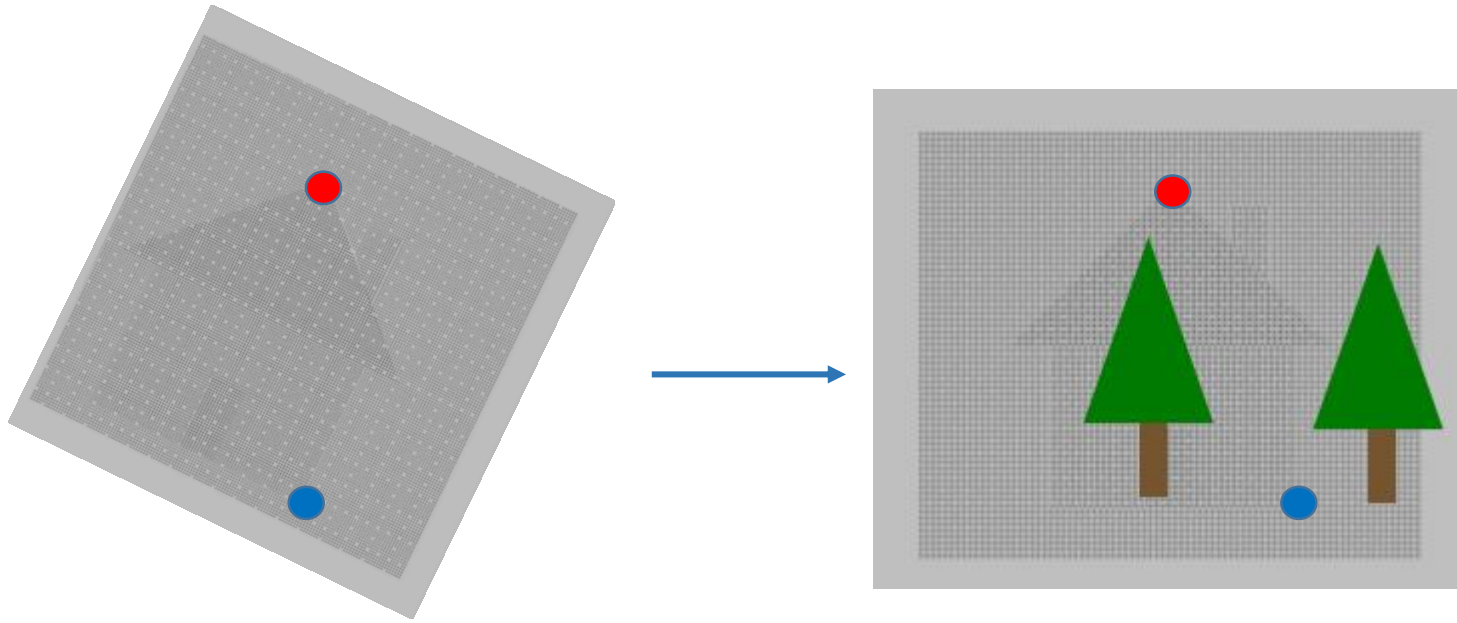
- Строится пирамида для каждого из 2х изображений
- Ищется преобразование на самом низком уровне
- Используется как начальное приближение для уточнения на следующих уровнях

Перекрывание объекта



- Как быть, если объект («дом») частично перекрыт другими объектами («ёлками»)
- Положение «ёлок» относительно дома разное на разных ракурсах
- Прямое попиксельное сопоставление изображений может не дать хорошего результата

Локальные особенности



- Найти хорошо различимые точки на изображениях («особенности» «локальный особые точки» «характеристические точки»)
- Определить, какой точке на одном изображении соответствуют какая точка на втором изображении
- Найти такое преобразование, которое совмещает найденные точки

Применение



Сопоставление изображений и трёхмерная
реконструкция

Применение

Motorbikes



Airplanes



Faces



Cars (Side)



Cars (Rear)



Spotted Cats



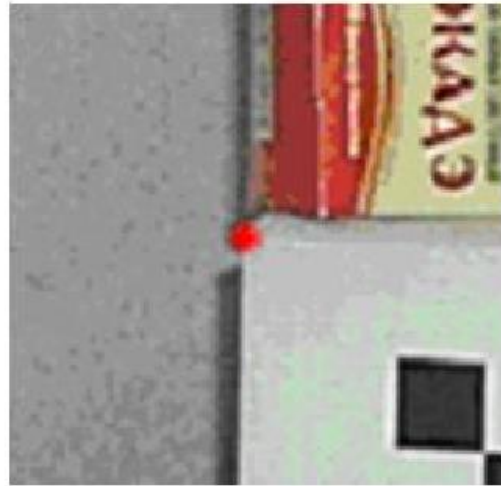
Классификация изображений и выделение объектов

Применение

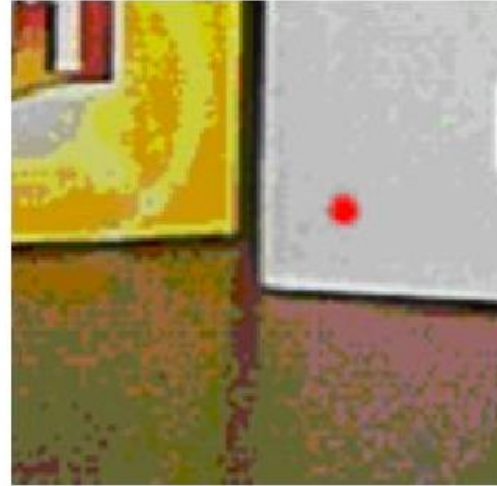


Поиск изображений по содержанию в базе изображений

Локальные особенности



Пример особой
точки



Пример точки, не
являющейся особой

Локальная (особая) точка изображения - это точка с характерной (особой) окрестностью, т.е. отличающаяся от всех других точек в некоторой окрестности.

Процесс определения достигается путём использования детектора и дескриптора.

Детектор – это метод извлечения особых точек из изображения.

Дескриптор – идентификатор особой точки, выделяющий её из остального множества особых точек.

Требования к особенностям

Повторимость (Repeatability)

- Особенность находится в том же месте сцены не смотря на изменения точки обзора и освещения

Локальность (Locality)

- Особенность занимает маленькую область изображения, поэтому работа с ней нечувствительна к перекрытиям

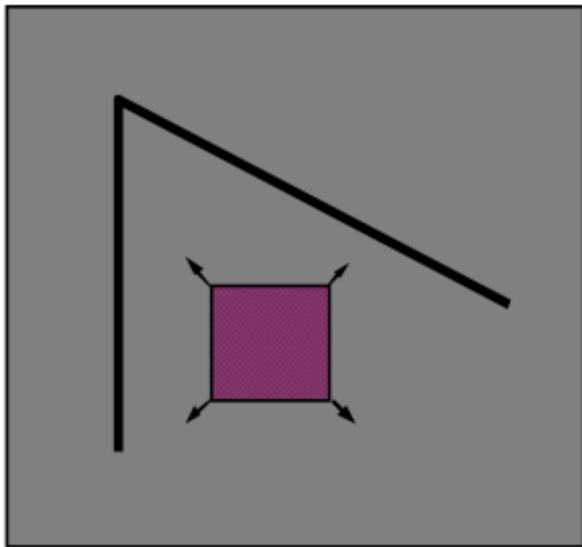
Значимость (Saliency)

- Каждая особенность имеет уникальное (distinctive) описание

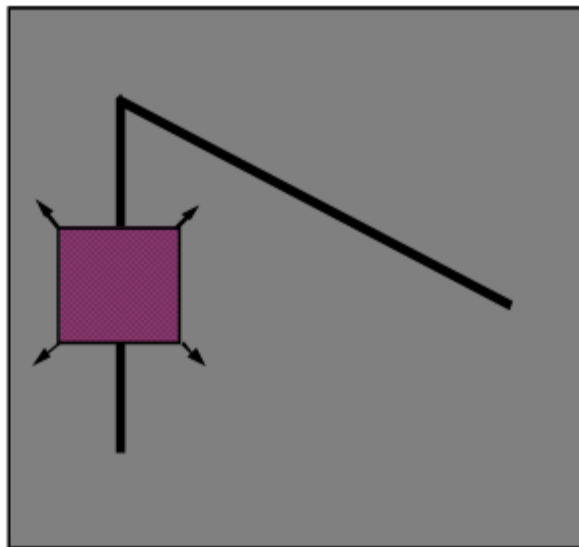
Компактность и эффективность

- Количество особенностей существенно меньше числа пикселей изображения

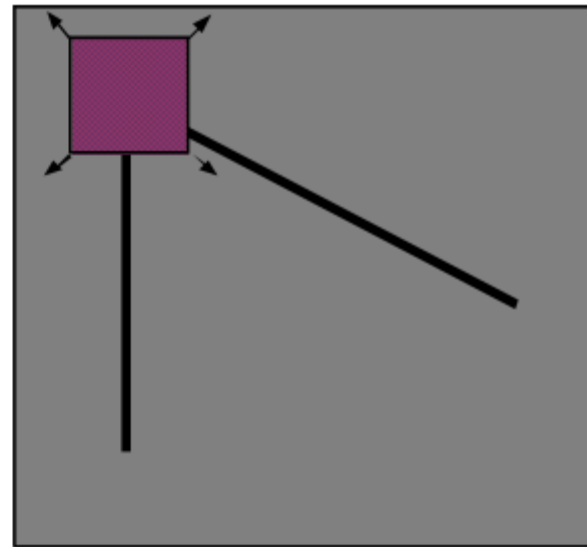
Локальные особенности



монотонный регион:
в любом направлении
изменений нет



«край»:
вдоль края
изменений нет



«уголок»:
изменения при
перемещении
в любую сторону

Детекторы углов

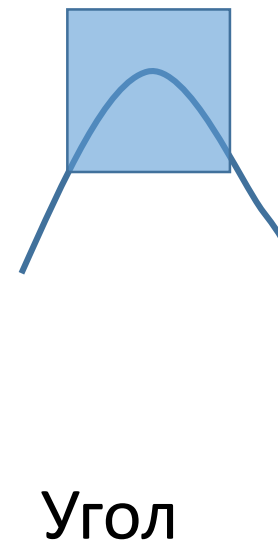
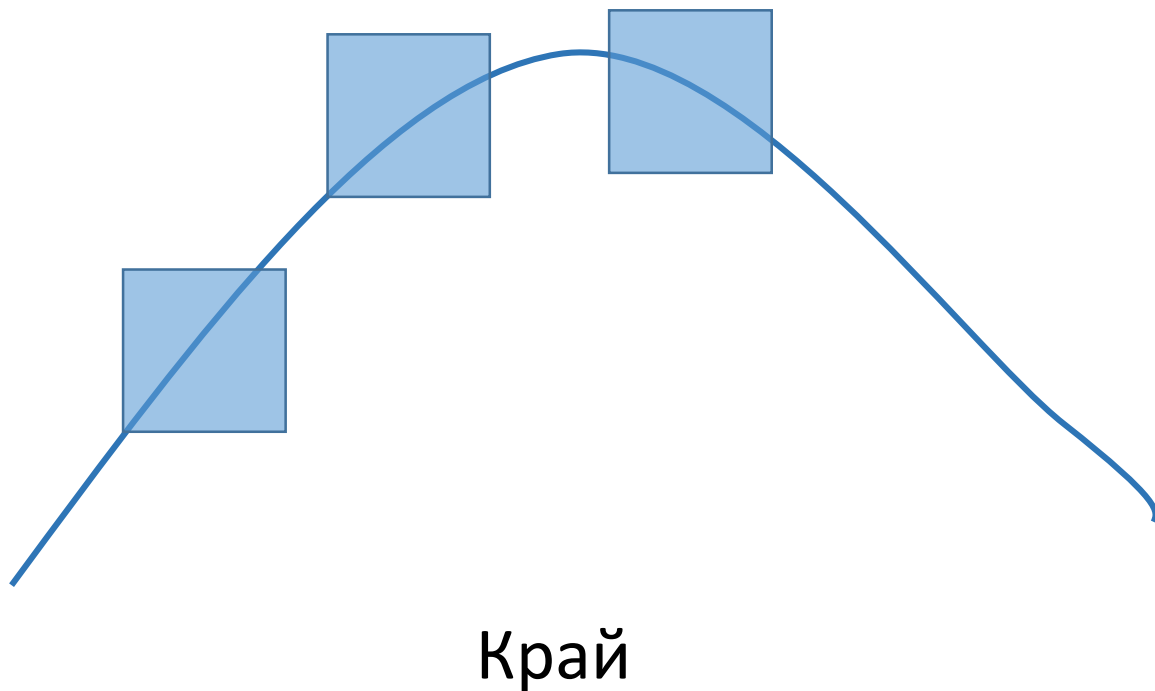
Работа с использованием особых точек началась с детектора Моравеца (Moravec, 1977). **Детектор Моравеца** – самый простой из существующих. Автор рассматривает изменение яркости квадратного окна W (обычно размера 3×3 , 5×5 , 7×7 пикселей) относительно интересующей точки при сдвиге окна W на 1 пиксель в 8-ми направлениях.

Харрис и Стефенс улучшили детектор Моравеца (Moravec)(1988), они рассмотрели производные яркости изображения для исследования изменений яркости по множеству направлений. **Детектор Харриса** является наиболее оптимальным детектором углов и широко применяется.

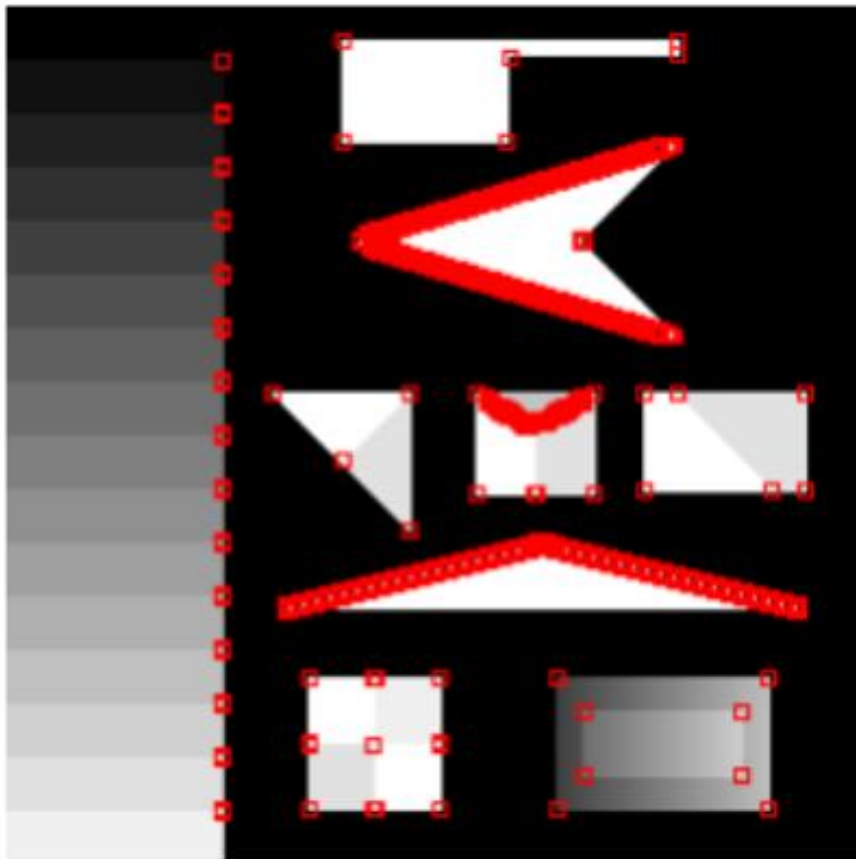
Детектор Харриса инвариантен к поворотам, частично инвариантен к аффинным изменениям интенсивности. К недостаткам стоит отнести чувствительность к шуму и зависимость детектора от масштаба изображения (для устранения этого недостатка используют многомасштабный детектор Харриса (multi-scale Harris detector)).

Масштабирование

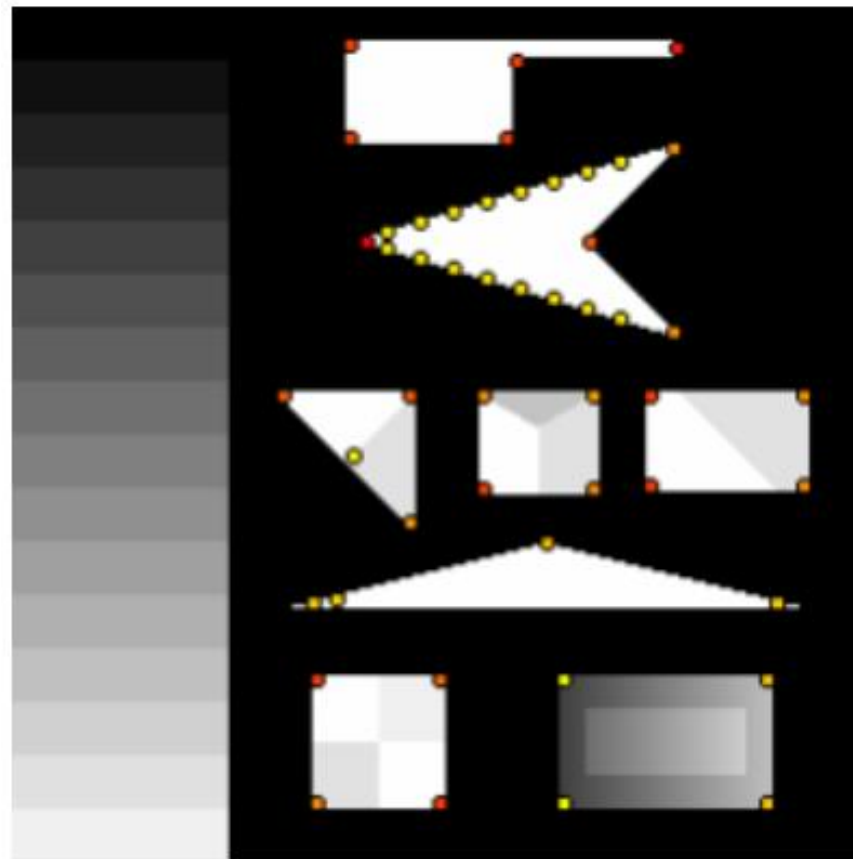
Детектор Харриса инвариантен к поворотам, частично инвариантен к аффинным изменениям интенсивности освещения.



Результат работы детекторов

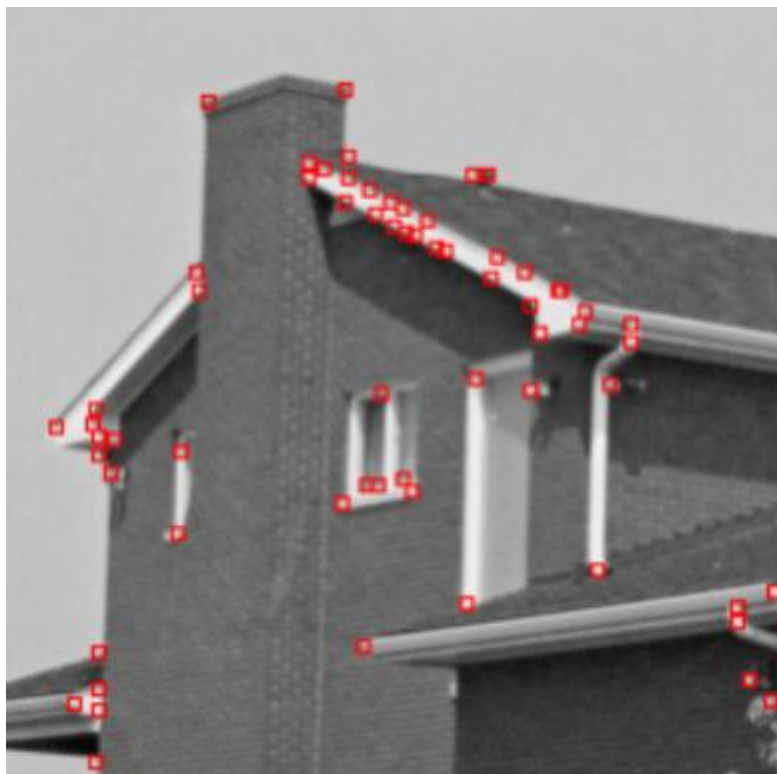


Детектор Моравеца

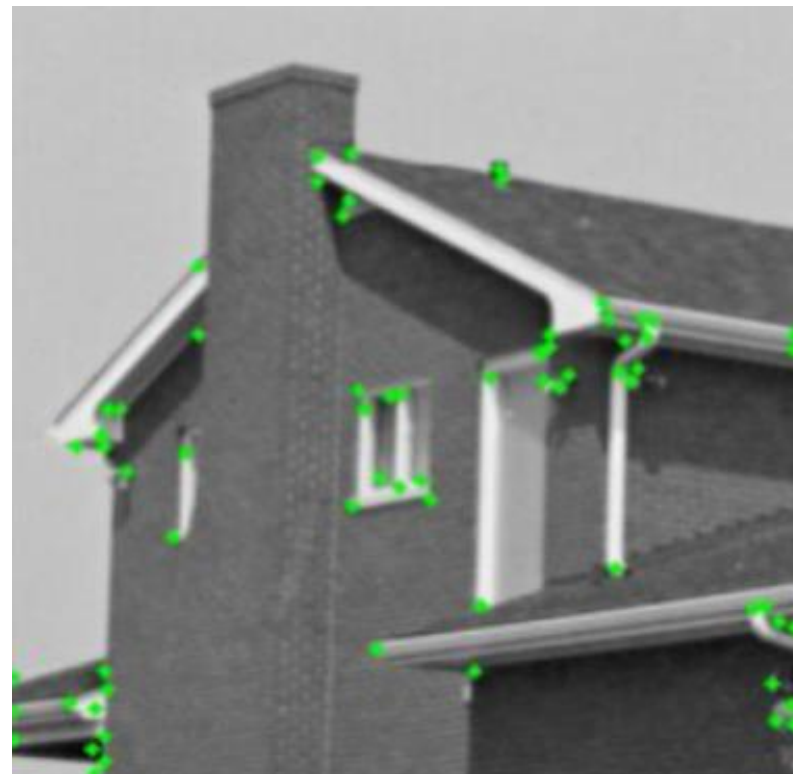


Детектор Харриса

Результат работы детекторов



Детектор Моравеца



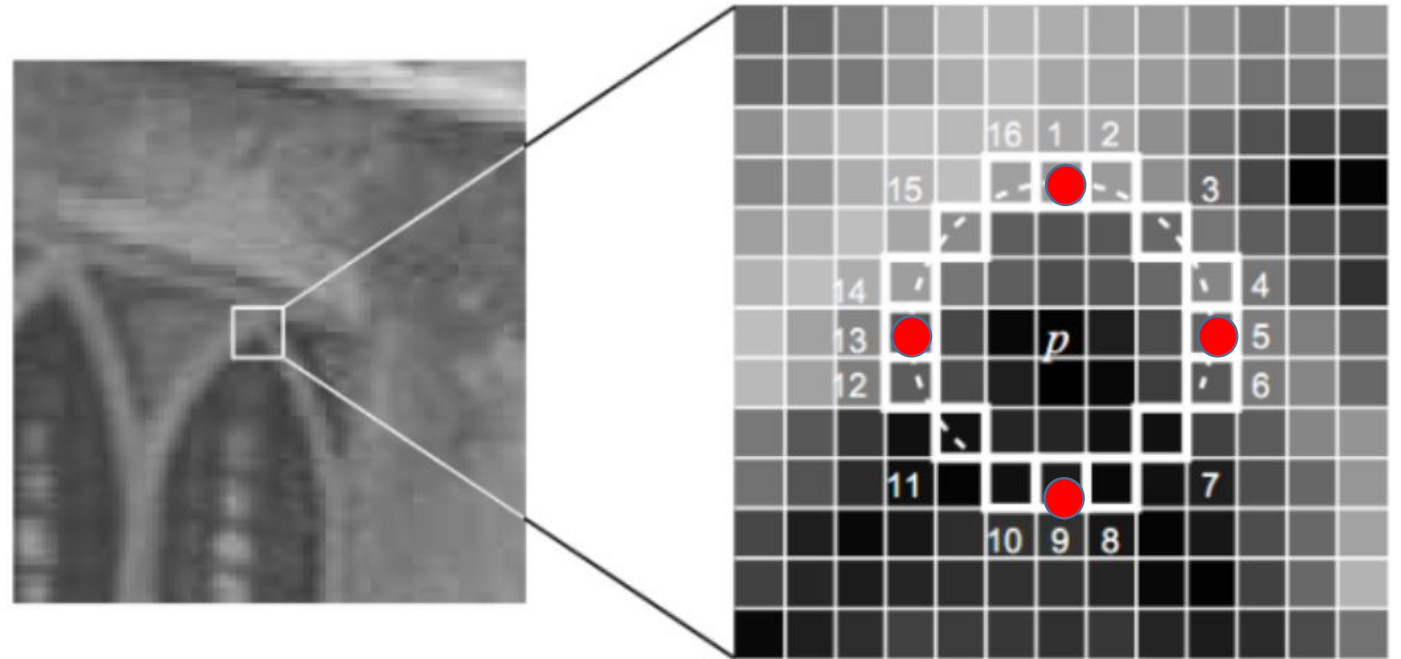
Детектор Харриса

Алгоритм FAST

В середине 2000-х годов появились алгоритмы быстрого поиска точек интереса. Наиболее ярким представителем данного класса алгоритмов является алгоритм **FAST** (*features from accelerated segment test* — особенности, полученные из ускоренной проверки сегментов).

В алгоритме рассматривается окружность из 16 пикселей. Яркость пикселей, лежащих на окружности, сравнивается с яркостью центральной точки и на основании ряда проверок принимается решение, является ли центральная точка характерной.

Последовательность проверок и их общее число подбираются и оптимизированы заранее на основе обширной обучающей выборки изображений.



Алгоритм FAST хорошо зарекомендовал себя в приложениях, осуществляющих слежение за объектами в реальном времени.

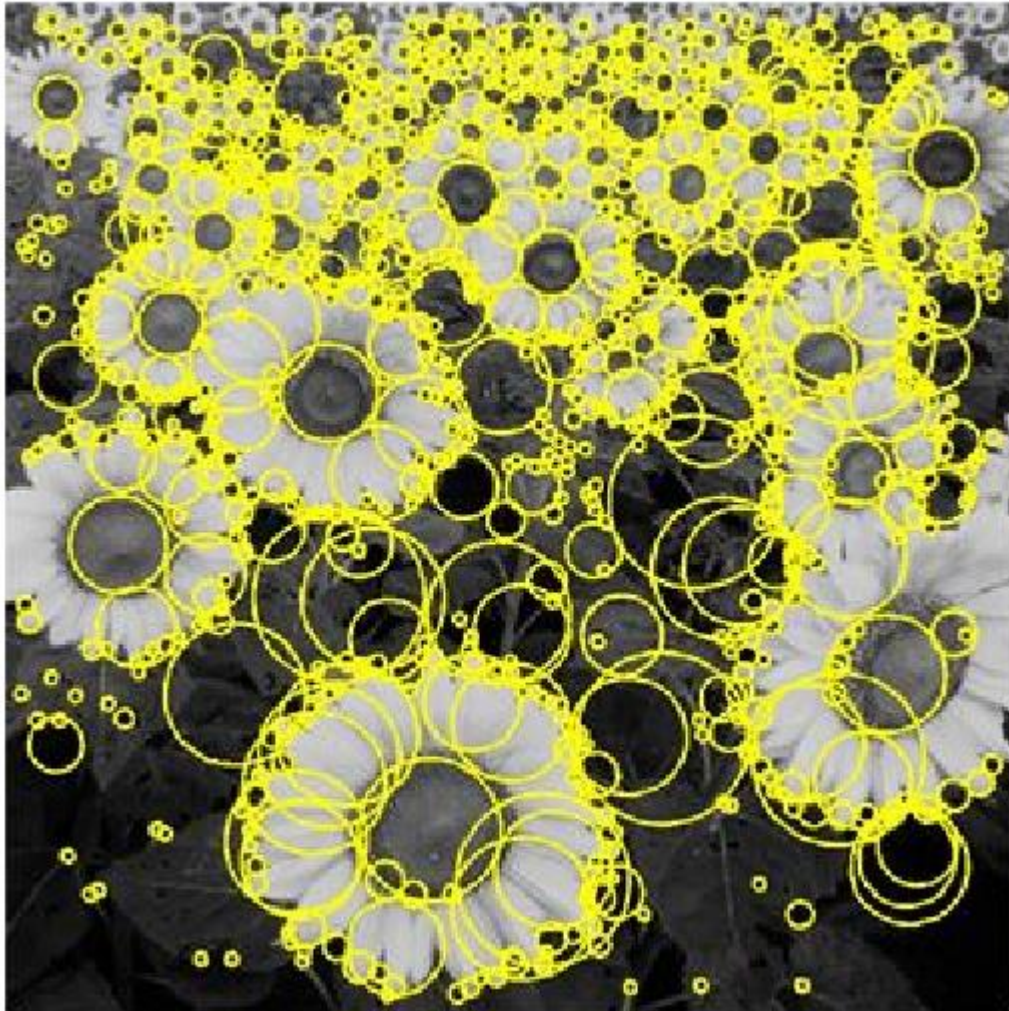
Инвариантность к масштабированию



Цель:

- определять размер окрестности особой точки в масштабированных версиях одного и того же изображения
- требуется метод выбора размера характеристической окрестности

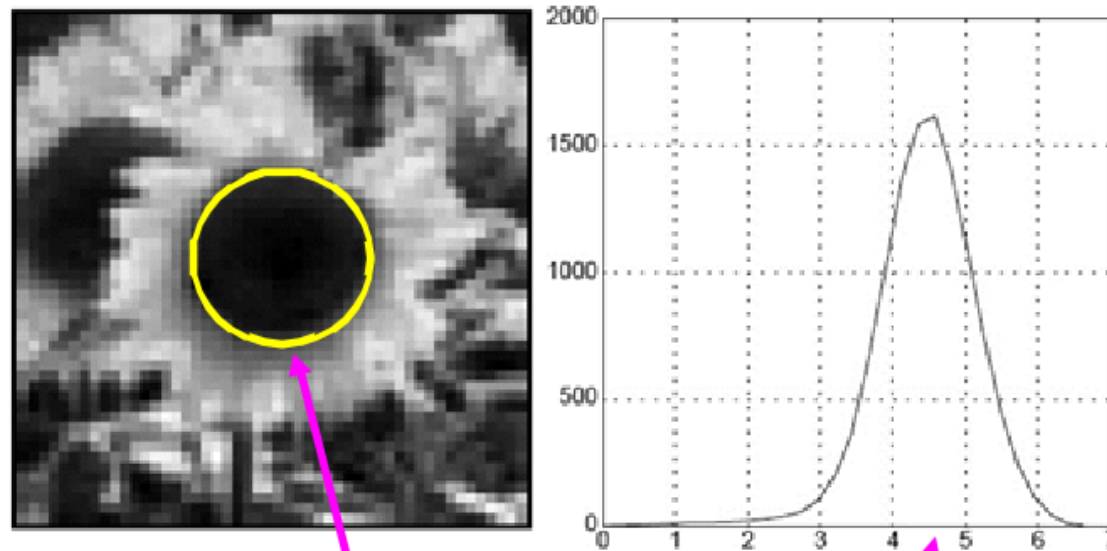
Блобы



Первый такой метод был предложен в конце 90-х гг, но находил он не углы а «блобы».

Характеристический размер

Характеристический размер определяется как масштаб, на котором достигается максимум отклика Лапласиана



Характеристический масштаб

DoG (Difference of Gaussia)

Разность гауссианов (Difference of Gaussian, DoG). Гауссианом (или изображением, размытым гауссовым фильтром) является изображение

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

Здесь L — значение гауссиана в точке с координатами (x, y) , а σ — радиус размытия. G — гауссово ядро, I — значение исходного изображения, $$ — операция свертки.*

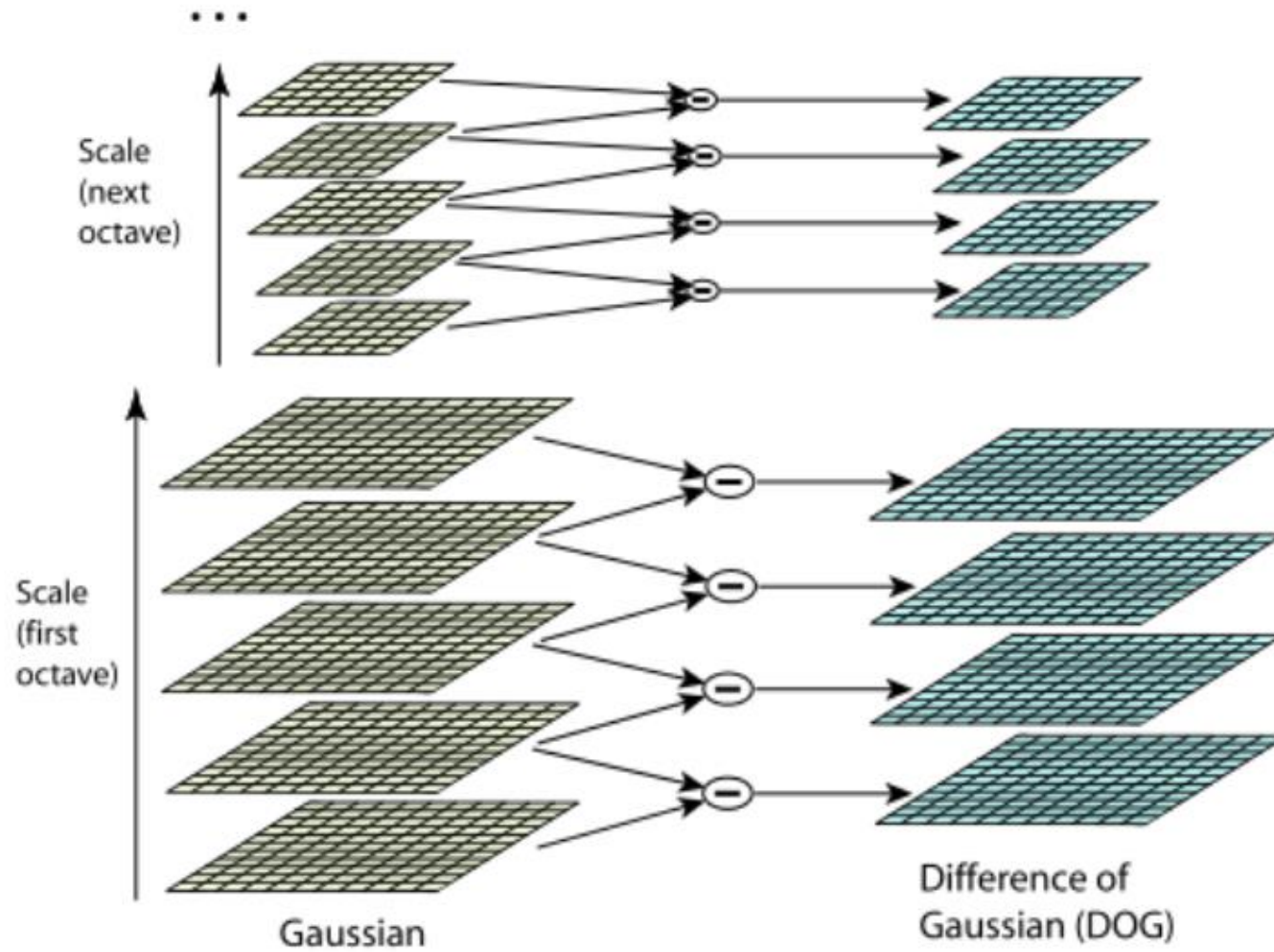
Разностью гауссианов называют изображение, полученное путем попиксельного вычитания одного гауссиана исходного изображения из гауссиана с другим радиусом размытия.

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma). \end{aligned}$$

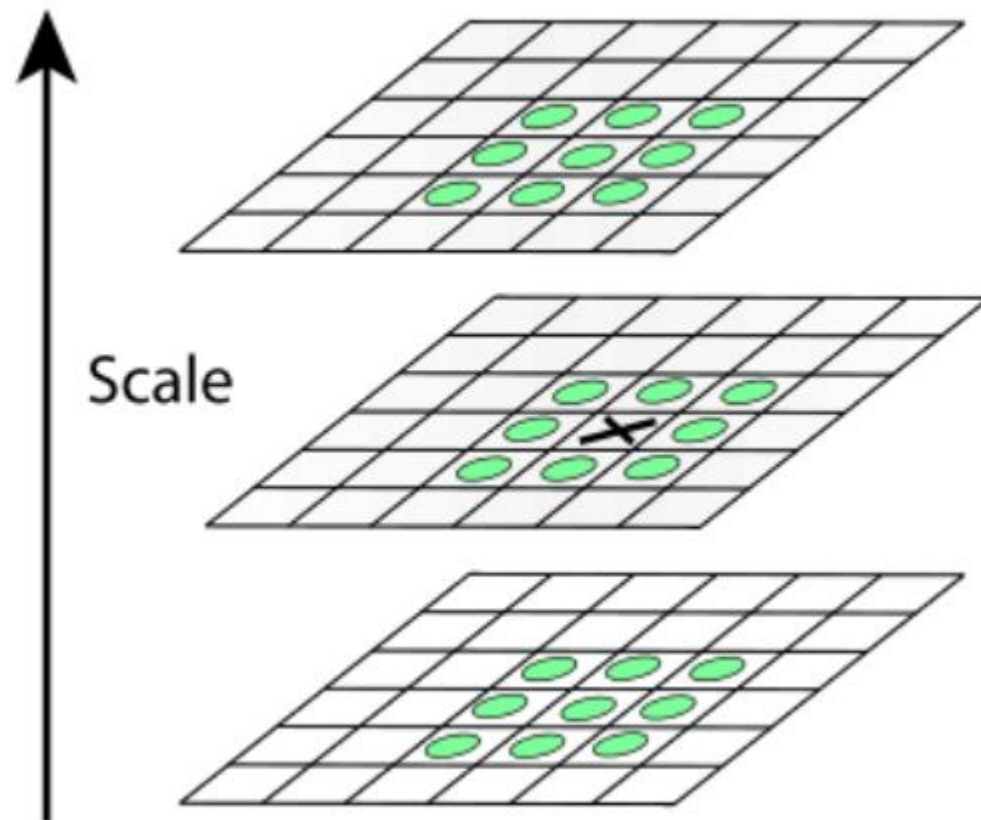
Различная степень размытия изображения гауссовым фильтром может быть принята за исходное изображение, взятое в некотором масштабе.

Инвариантность относительно масштаба достигается за счет нахождения ключевых точек для исходного изображения, взятого в разных масштабах. Для этого строится пирамида гауссианов

DoG (Difference of Gaussia)



DoG (Difference of Gaussia)

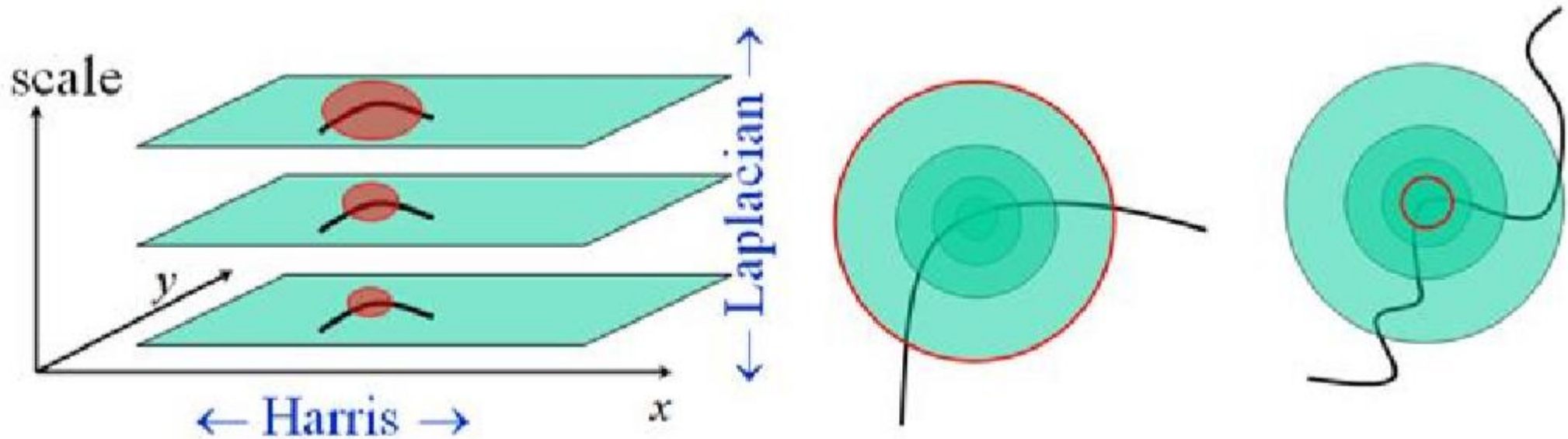


Будем считать точку особой, если она является локальным экстремумом разности гауссианов.

В каждом изображении из пирамиды DoG ищутся точки локального экстремума. Каждая точка текущего изображения DoG сравнивается с её восемью соседями и с девятью соседями в DoG, находящихся на уровень выше и ниже в пирамиде. Если эта точка больше (меньше) всех соседей, то она принимается за точку локального экстремума.

Детектор Harris-Laplacian

Разные варианты чередования вычисления функции Харриса и Лапласиана позволяют выделить углы на изображении, но с характеристическим размером



Детекторы областей

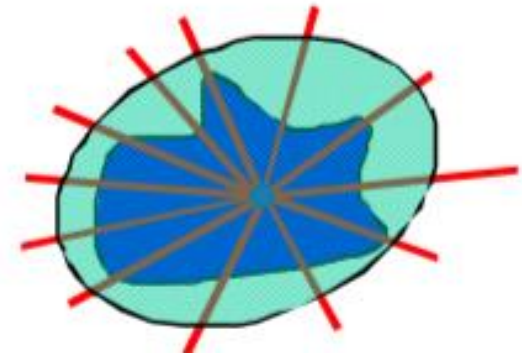
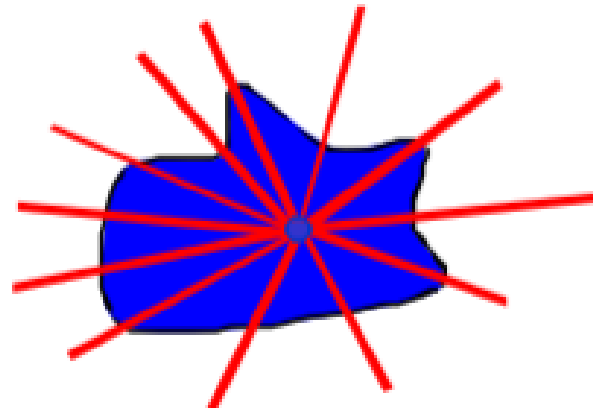
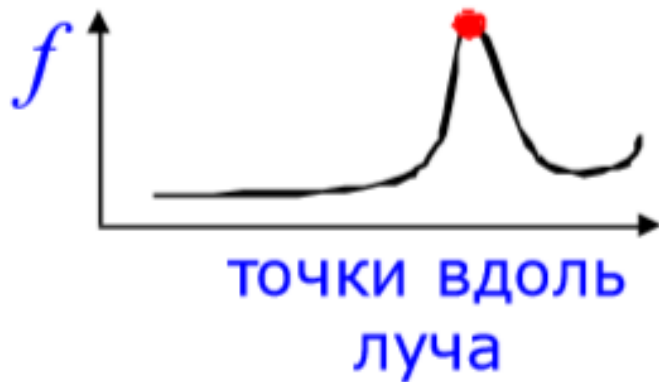
Можно попробовать работать с более уникальными характеристиками изображения – с **областями**. Экстремальных областей гораздо меньше, но они более точно характеризуют сцену или объект.

У экстремальных (особых) областей в этом контексте есть два важных свойства:

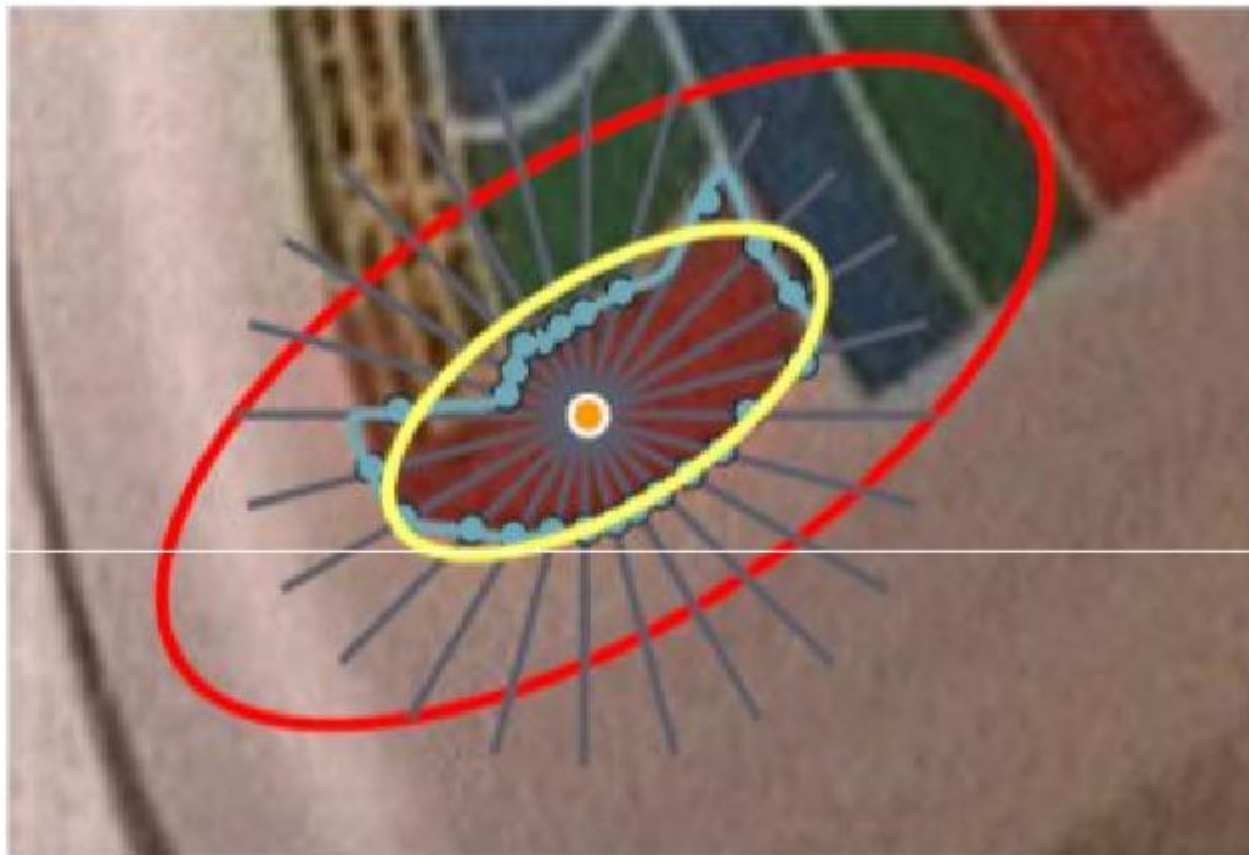
- непрерывное преобразование координат изображения. Это означает, что он является аффинно-инвариантным, и не имеет значения, будет ли изображение искажено или искажено.
- монотонное преобразование интенсивности изображения. Подход, конечно, чувствителен к естественным световым эффектам, таким как изменение дневного света или движущихся теней.

Детектор областей IBR (Intensity-extrema based regions)

- Идти от локального экстремума яркости по лучам, считая некоторую величину f
- Остановка при достижении пика f



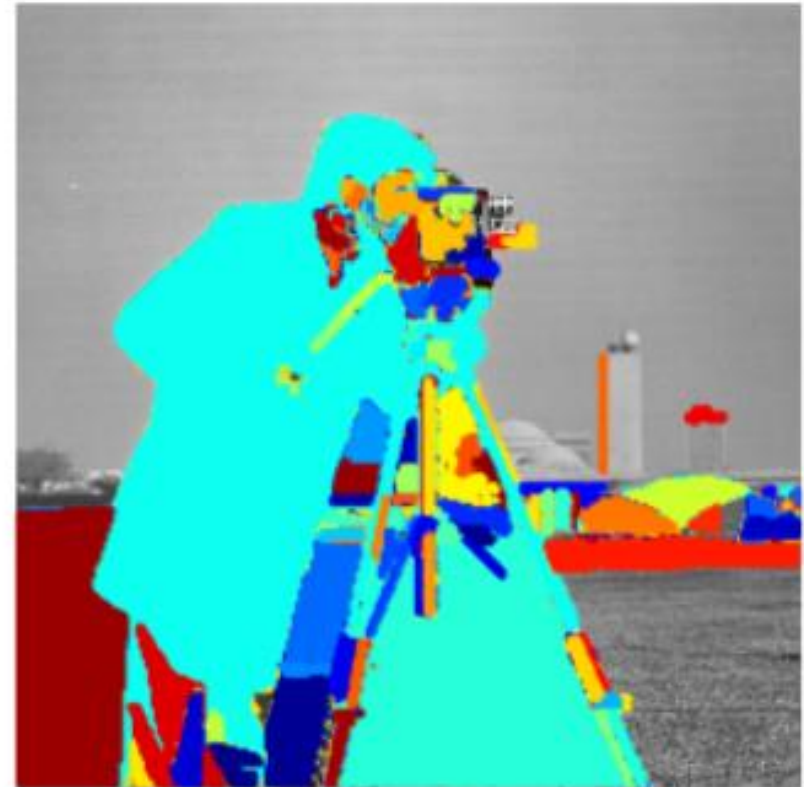
Детектор областей IBR (Intensity-extrema based regions)



MSER = Maximally Stable Extremal Regions

Метод находит соответствия между элементами изображения в двух изображениях с **разных точек зрения**.

- Задать порог яркости T
- Провести сегментацию
- Извлечь области
- Для каждой области найти порог, при котором рост площади минимален
- Описать вокруг области эллипс

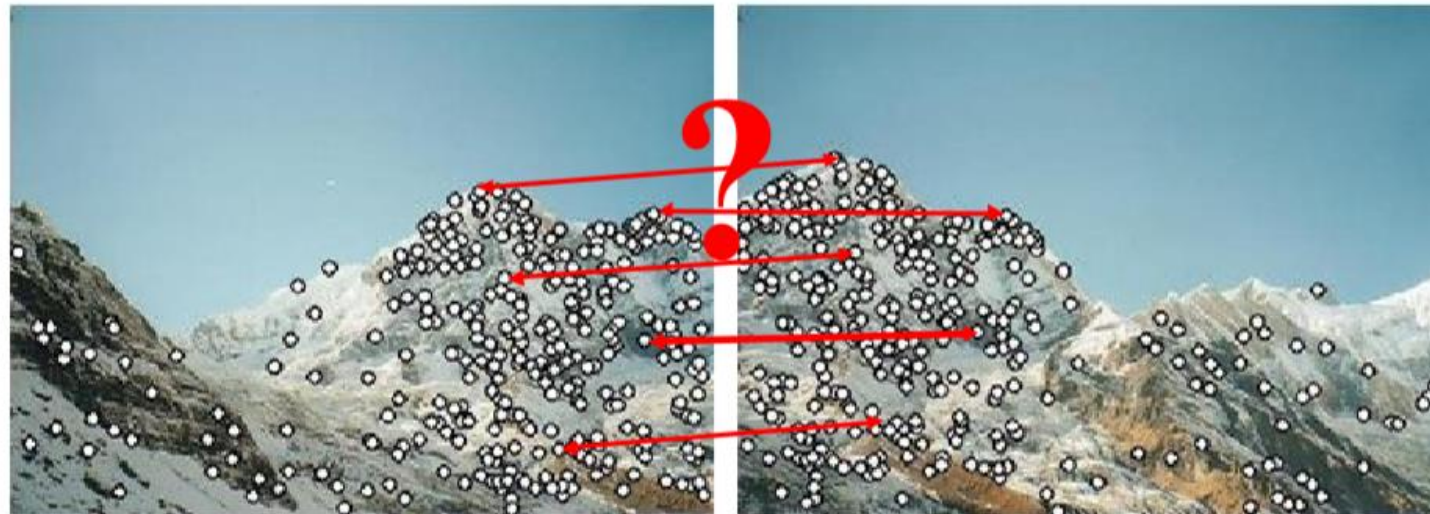


Резюме

- Локальные особенности – один из основных инструментов анализа изображений
- Рассмотрели основные детекторы особенностей:
 - Детекторы углов: Harris (Forstner), Harris-Laplace)
 - Детекторы Блобов: LoG (Laplacian of Gaussian), DoG (Difference of Gaussians)
 - Детекторы областей: IBR (Intensity-extrema based regions), MSER (Maximally Stable Extremal Regions)

Дескрипторы

Точки найдены, а как их отличить друг от друга



Дескриптор — идентификатор ключевой точки, выделяющий её из остальной массы особых точек.

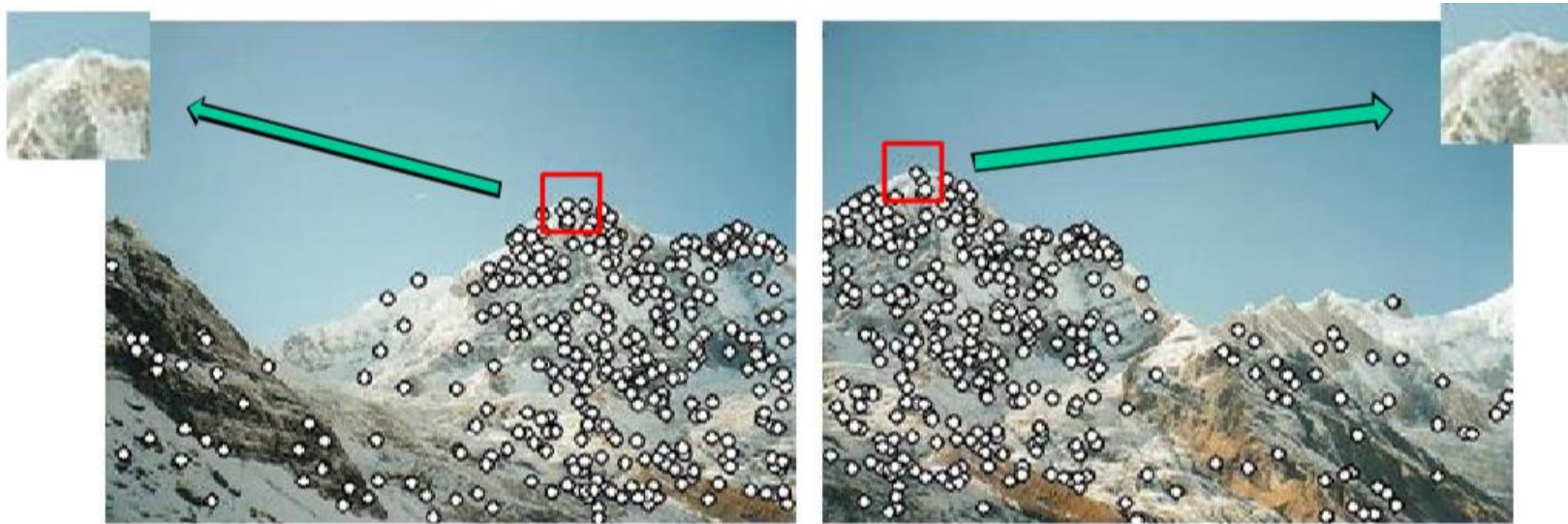
Дескриптор представляет собой запись фрагмента картинки в числовом виде, с помощью дескрипторов можно достаточно точно сравнивать фрагменты без использования самих фрагментов, затем проводится кластеризация, т.е. распределение похожих дескрипторов по кластерам

Дескрипторы

Дескрипторы должны быть:

- специфичны (различные для разных точек)
- локальны (зависеть только от небольшой окрестности)
- инвариантны (к искажениям, изменениям освещенности)
- просты в вычислении

Дескрипторы. Простейший подход



- Возьмём квадратные окрестности, со сторонами, параллельными строкам и столбцам изображения
- Яркости пикселей будут признаками
- Сравнивать будем как изображения попиксельно (SAD, SSD)
- Такая окрестность инвариантна только к сдвигу изображения

Дескрипторы. Инвариантность к яркости

- Можем добиться следующим образом:
 - Локальная нормализация гистограммы
 - Дескрипторы, основанные на градиенте яркости, инвариантны к сдвигу яркости
 - Нормирование яркости - вычесть среднее значение, поделить на дисперсию

$$I' = (I - \mu) / \sigma$$



нормализация

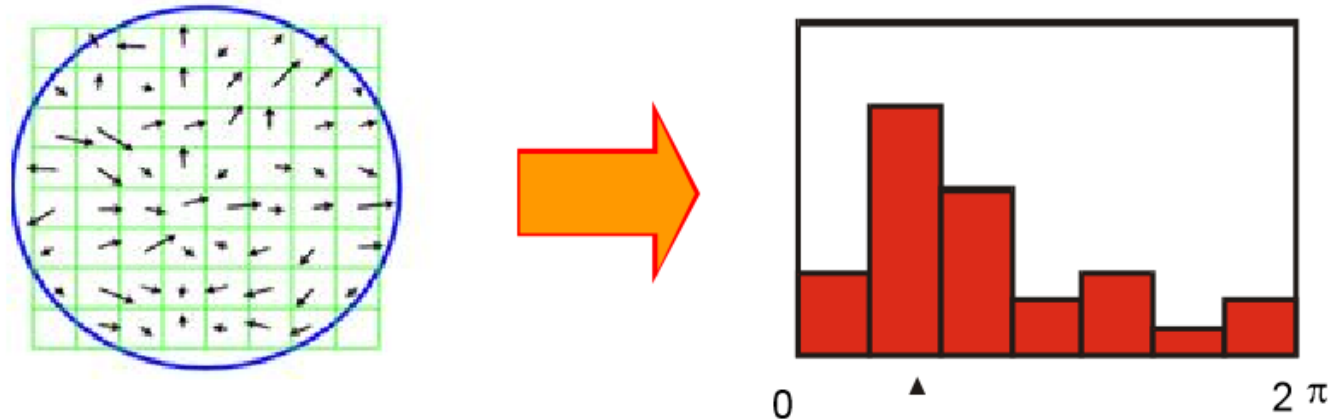


Дескрипторы. SIFT

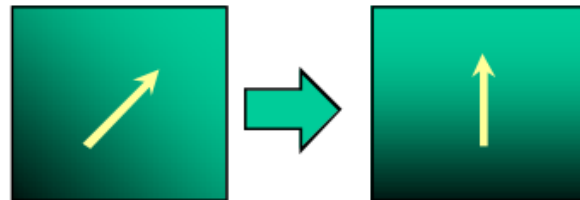
- Scale-Invariant Feature Transform:
 - Детектор DoG
 - Определение положения и масштаба особенности
 - Ориентация
 - Определение доминантной ориентации по градиентам
 - Дескриптор
 - Использование статистик по направлению градиентам

SIFT. Ориентация

- Идея: найти основное (доминантное) направление градиентов пикселей в окрестности точки
- Посчитаем гистограмму, взвешивая вклад по гауссиане с центром в точке

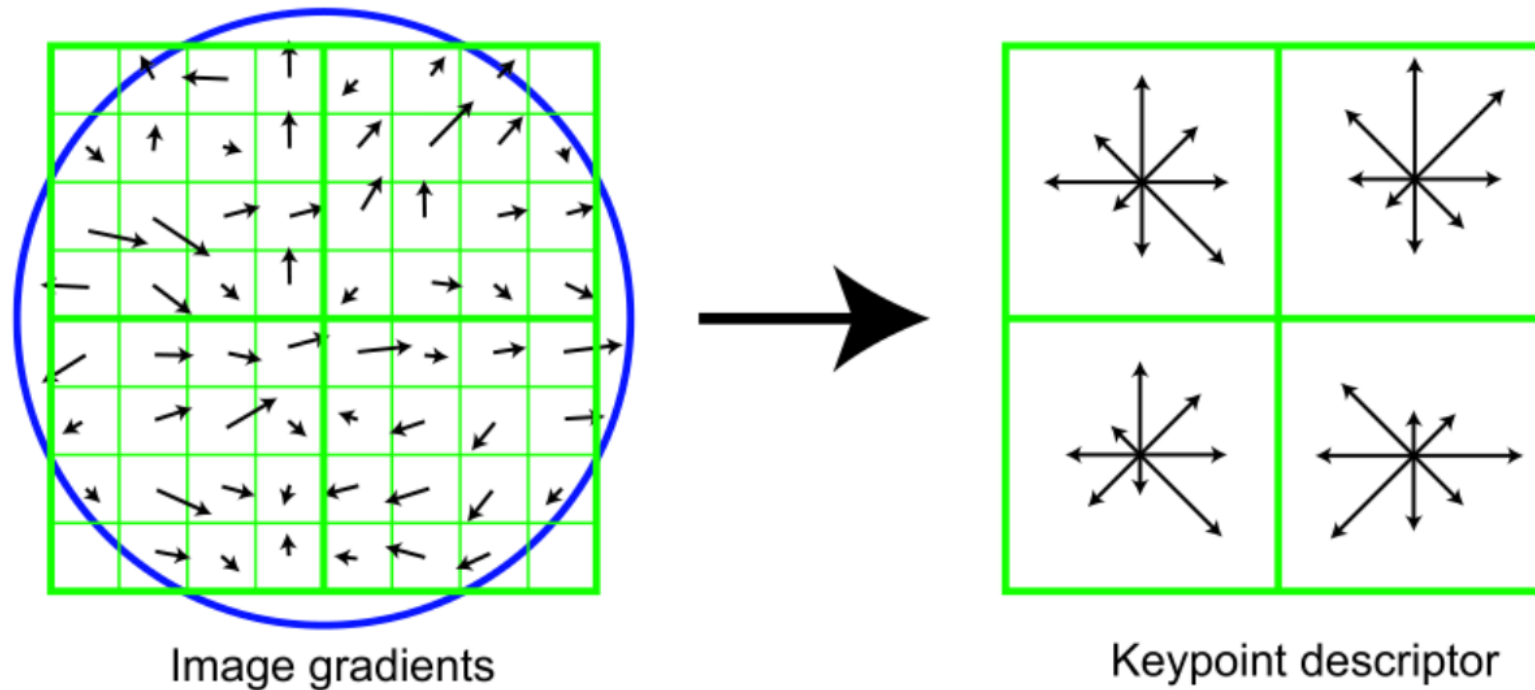


- Повернуть фрагмент так, чтобы доминантное направление градиента было направлено вверх



- Если локальных максимумов несколько – считаем, что несколько точек с разной ориентацией

SIFT. Дескриптор



- Для учета локальных свойств разделим окрестность на блоки сеткой, в каждом блоке посчитаем свою гистограмму градиентов
- Обычно – сетка 4x4, в каждой гистограмма с 8ю ячейками
- Стандартная длина вектора-дескриптора – 128 ($4 \cdot 4 \cdot 8$)
- Сравниваем как вектор (разные метрики)

Резюме SIFT

- Детектор SIFT весьма специфичен, устойчив к изменениям освещения, небольшим сдвигам
- Вся схема SIFT (детектор, выбор окрестностей, дескриптор) оказалась очень эффективным инструментом для анализа изображений
- Очень широко используется

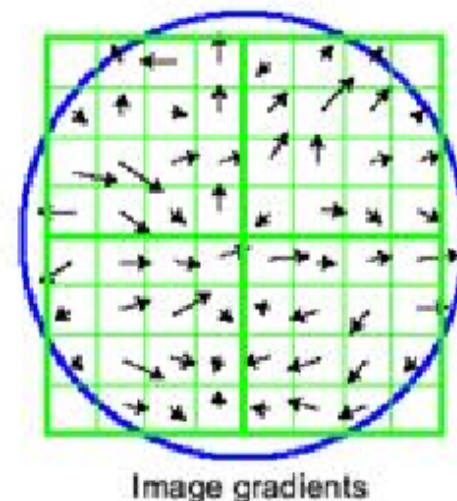


Image gradients

Классификация по признакам

