

Введение в фотограмметрию

Локальные ключевые точки SIFT



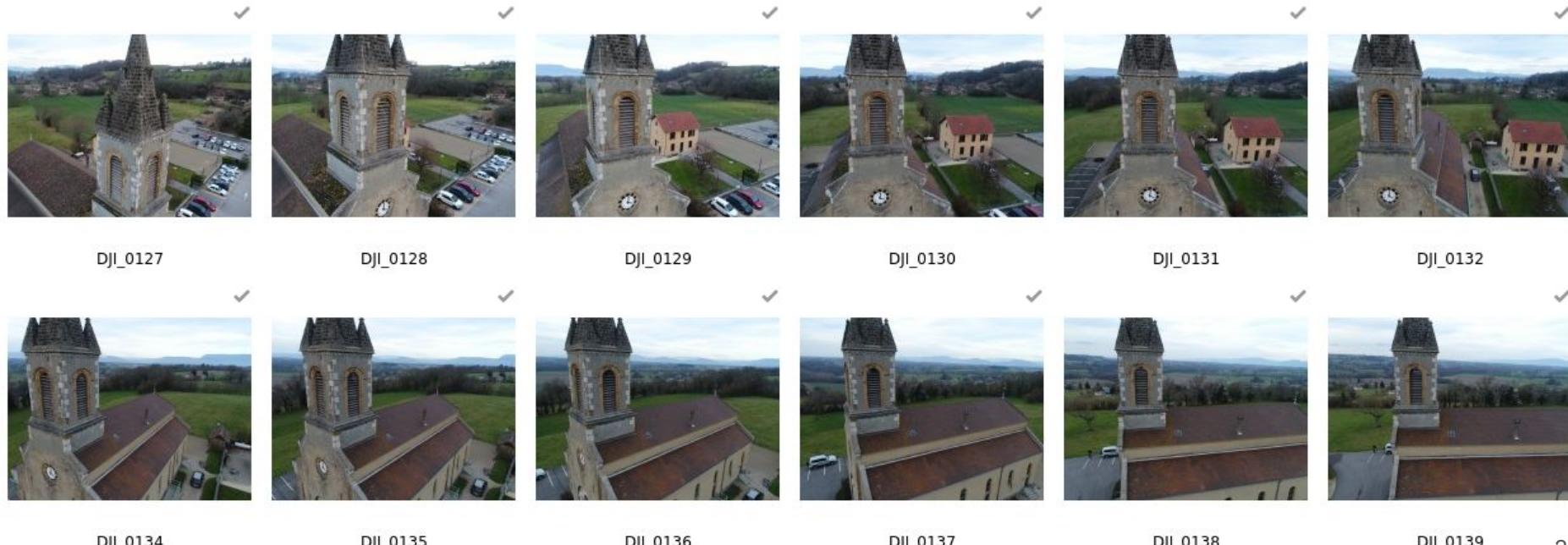
Фотограмметрия. Лекция 1

- План курса
- Ключевые точки
- SIFT детектор
- SIFT дескриптор

Полярный Николай
polarnick239@gmail.com

Введение в фотограмметрию: 3D съемка

По множеству фотографий одной и той же сцены получить цифровую модель



Данные предоставил Stéphane Prodent

Полигональная 3D модель

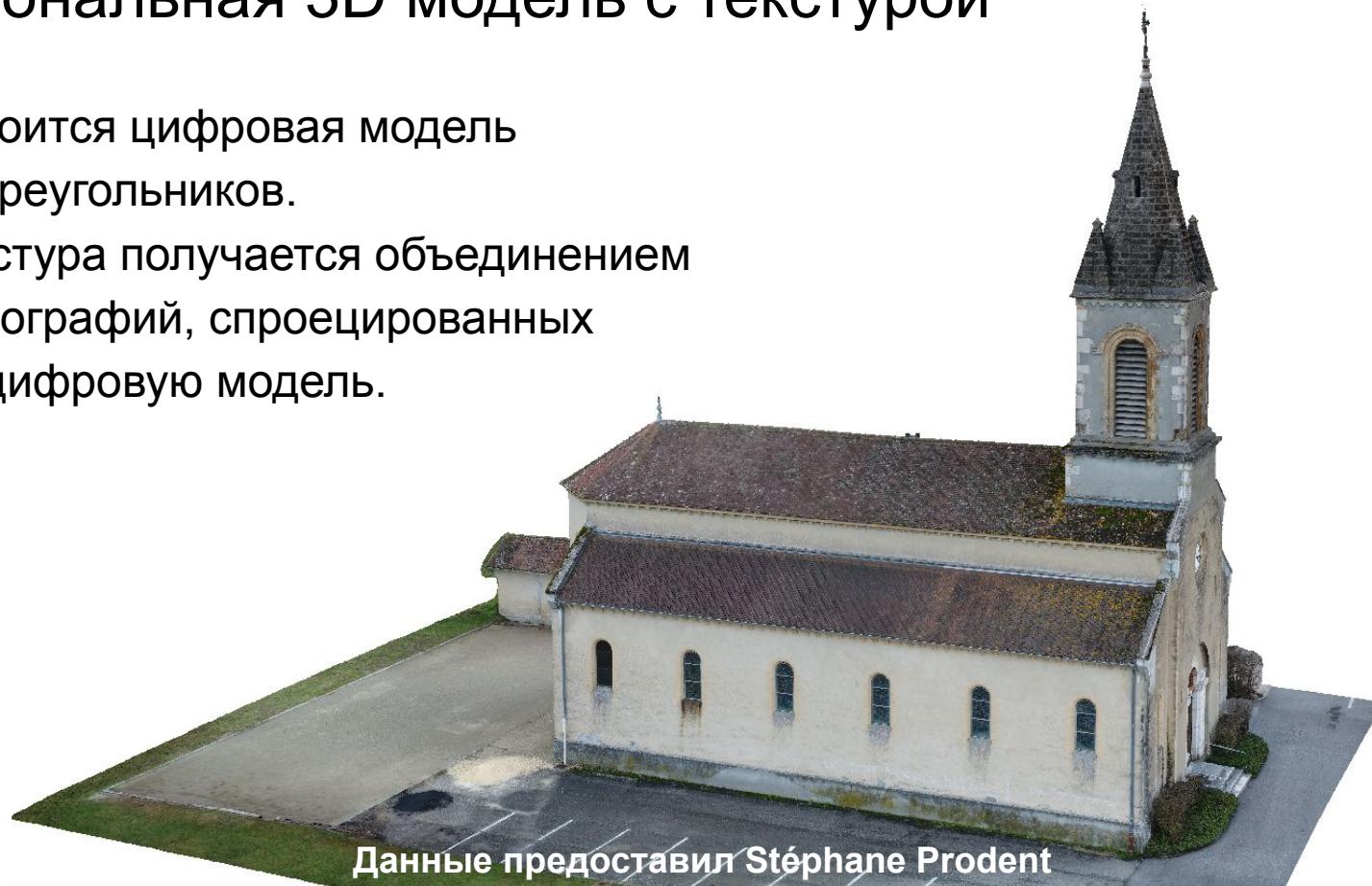
- 1) Строится цифровая модель из треугольников.



Данные предоставил Stéphane Prodent

Полигональная 3D модель с текстурой

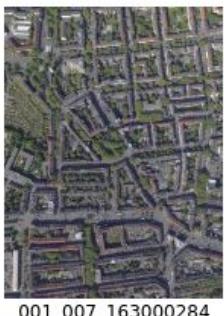
- 1) Строится цифровая модель из треугольников.
- 2) Текстура получается объединением фотографий, спроектированных на цифровую модель.



Данные предоставил Stéphane Prodent

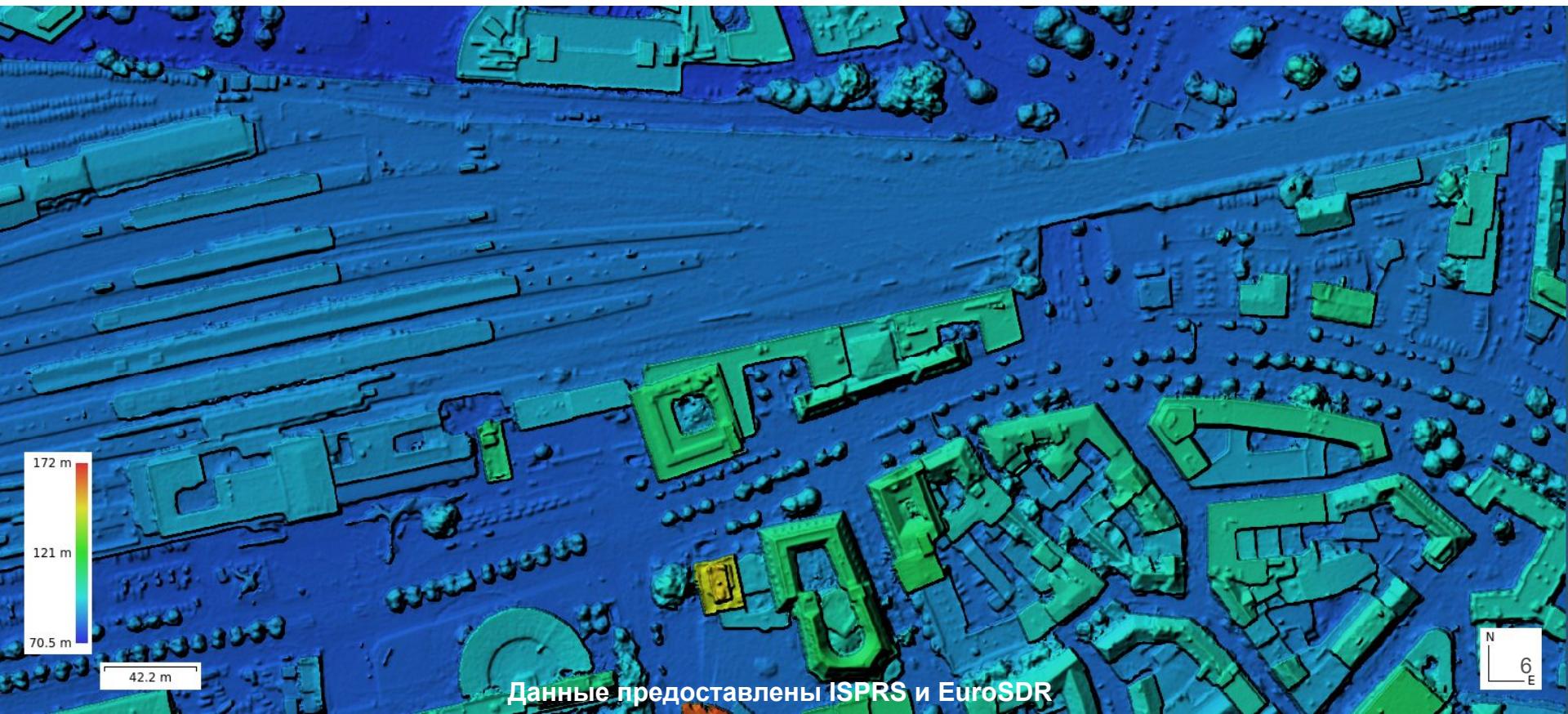
Введение в фотограмметрию: 2.5D съемка

Пример аэрофотосъемки города:



Данные предоставлены ISPRS и EuroSDR

DEM - Digital Elevation Model



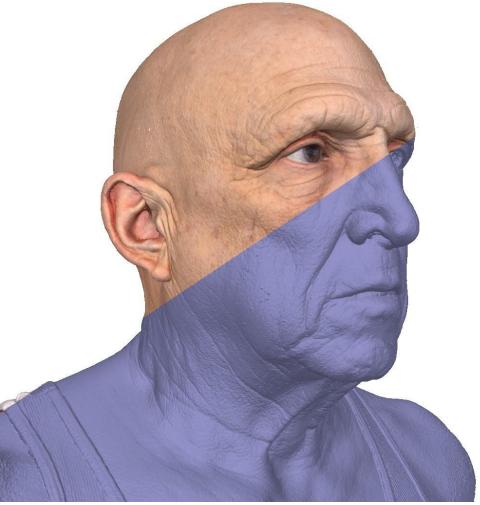
Ортомозаика - объединение проекций фотографий на DEM

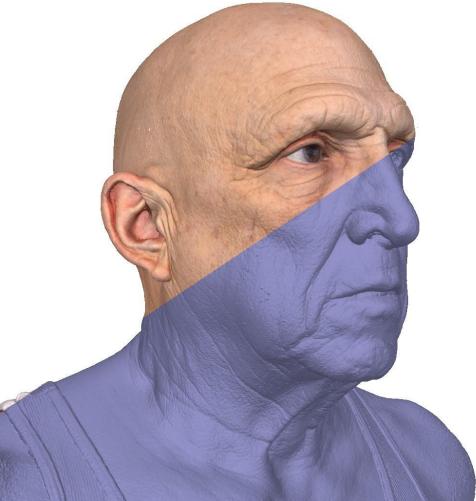


42.2 m

N
E
7

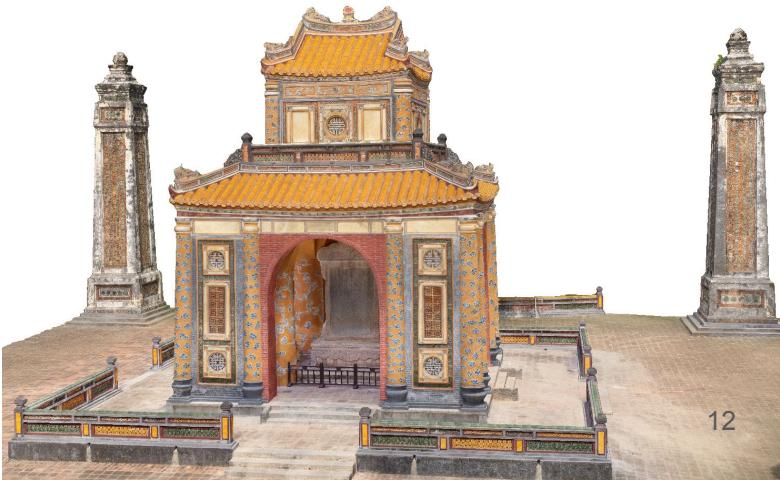
Данные предоставлены ISPRS и EuroSDR

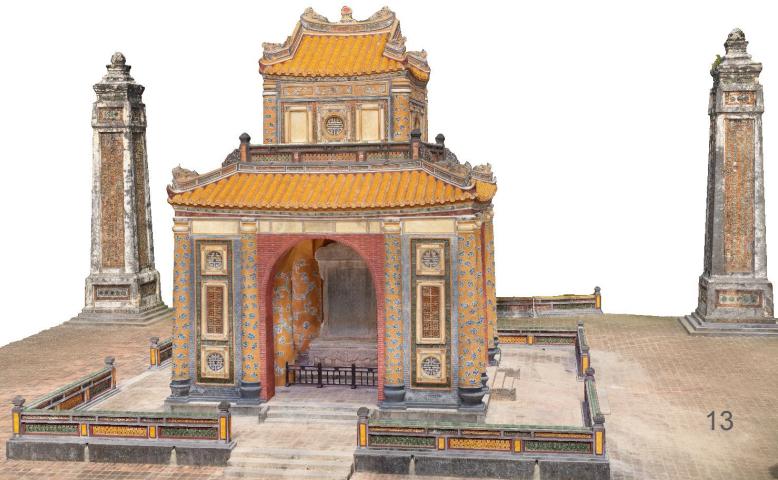












 Agisoft
Metashape

Программа курса по неделям

- 1) 14.02 [ДЗ] Ключевые точки (**SIFT**)
- 2) 21.02 [ДЗ] Сопоставление ключевых точек (**RANSAC, K-ratio test, ...**)
- 3) 28.02 [ДЗ] Определение взаимного расположения пары камер
- 4) 07.03 Определение взаимного расположения пары облаков точек снятых с LIDAR сканера
- 5) 14.03 Построение карт глубины (**Semi-Global Matching**)
- 6) 21.03 [ДЗ] Определение расположения всех камер (**Bundle Adjustment**)
- 7) 28.03 Обнаружение 3D проводов, построение облака точек, **Poisson** реконструкция 3D модели
- 8) 04.04 [ДЗ] Построение карт глубины (**Patch Match**)
- 9) 11.04 Реконструкция 3D модели (**TVL1, TGV, Out-of-Core**)
- 10) 18.04 [ДЗ] Реконструкция 3D модели (**Delaunay triangulation + Graph min-cut**)
- 11) 25.04 Текстурирование модели и построение ортомозаики
- 12) 02.05 Автоматическое планирование облета объекта (**3D Mission Planner**)

Программа курса по неделям

- 1) 14.02 [ДЗ] Ключевые точки (**SIFT**)
- 2) 21.02 [ДЗ] Сопоставление ключевых точек (**RANSAC, K-ratio test, ...**)
- 3) 28.02 [ДЗ] Определение взаимного расположения пары камер
- 4) 07.03 Определение взаимного расположения пары облаков точек снятых с LIDAR сканера
- 5) 14.03 Построение карт глубины (**Semi-Global Matching**)
- 6) 21.03 [ДЗ] Определение расположения всех камер (**Bundle Adjustment**)
- 7) 28.03 Обнаружение 3D проводов, построение облака точек, **Poisson** реконструкция 3D модели
- 8) 04.04 [ДЗ] Построение карт глубины (**Patch Match**)
- 9) 11.04 Реконструкция 3D модели (**TVL1, TGV, Out-of-Core**)
- 10) 18.04 [ДЗ] Реконструкция 3D модели (**Delaunay triangulation + Graph min-cut**)
- 11) 25.04 Текстурирование модели и построение ортомозаики
- 12) 02.05 Автоматическое планирование облета объекта (**3D Mission Planner**)

Важные моменты:

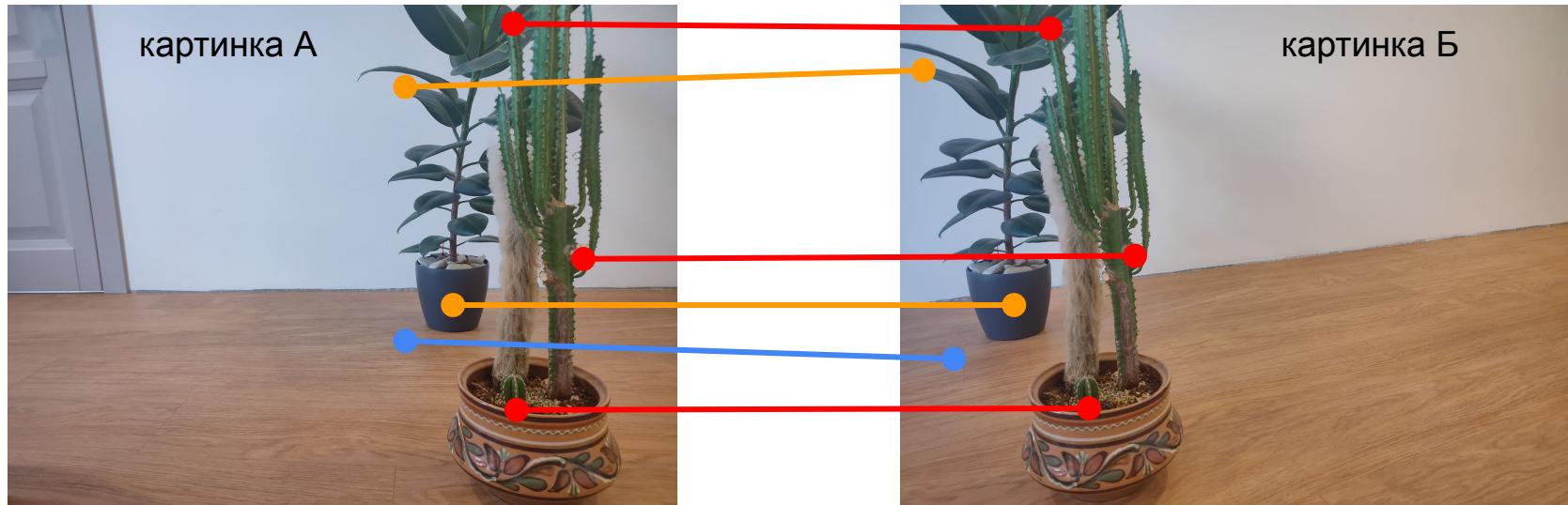
- 1) Чат в телеграмме (напишите мне и я вас добавлю - <https://t.me/PolarNick239>)
- 2) Записи будут выкладываться на **youtube**
- 3) Домашние задания будут на **github** - <https://github.com/PhotogrammetryCourse>

Задача сопоставления фотографий



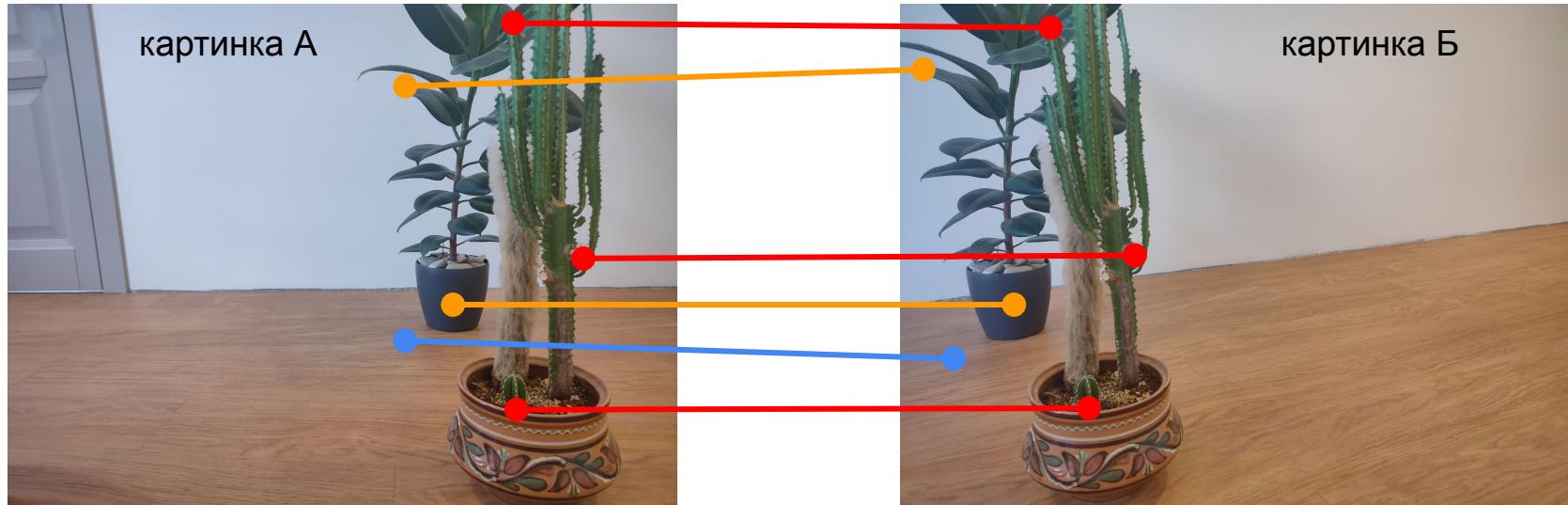
- 1) Как их сопоставить? Как алгоритму понять где одно и то же?

Задача сопоставления фотографий



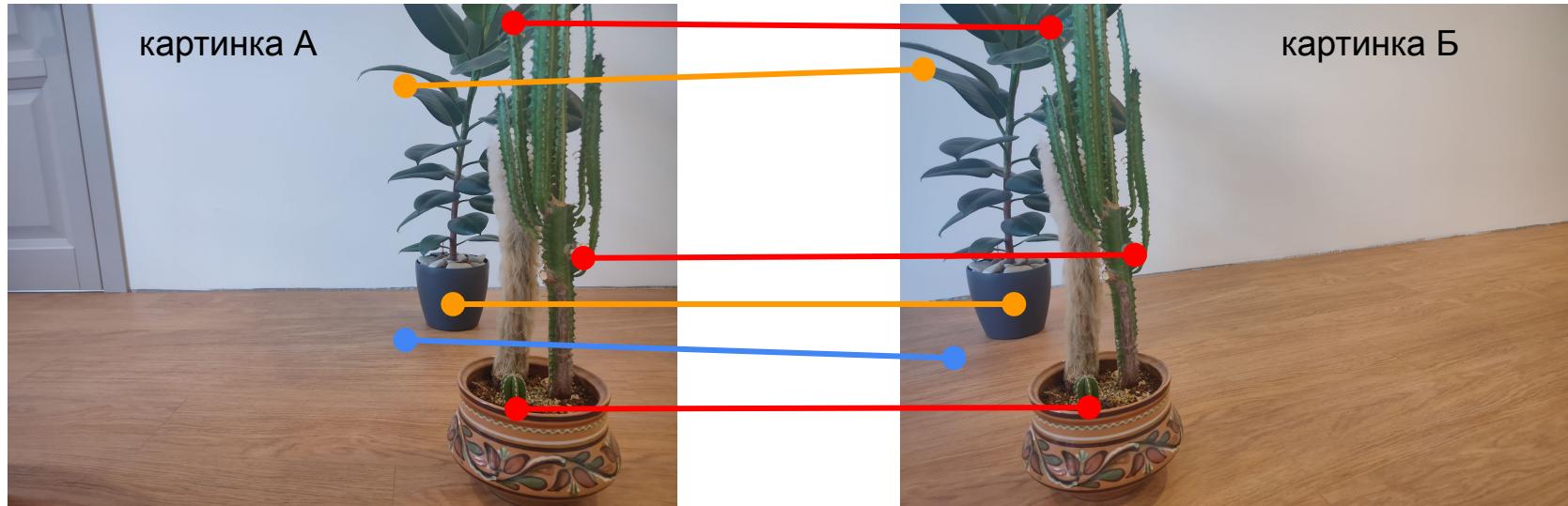
- 1) Как их сопоставить? Как алгоритму понять где одно и то же?
- 2) Если свести задачу к выбору ключевых точек - какие к ним требования?

Задача сопоставления фотографий



- 1) Как их сопоставить? Как алгоритму понять где одно и то же?
- 2) Если свести задачу к выбору ключевых точек - какие к ним требования?
- 3) К каким искажениям должны быть устойчивы (инвариантны) точки?

Задача сопоставления фотографий



- 1) Как их сопоставить? Как алгоритму понять где одно и то же?
- 2) Если свести задачу к выбору ключевых точек - какие к ним требования?
- 3) К каким искажениям должны быть устойчивы (инвариантны) точки?
- 4) Как сопоставить точки? Как оценить похожесть точек?

Key Points: Detector & Descriptor

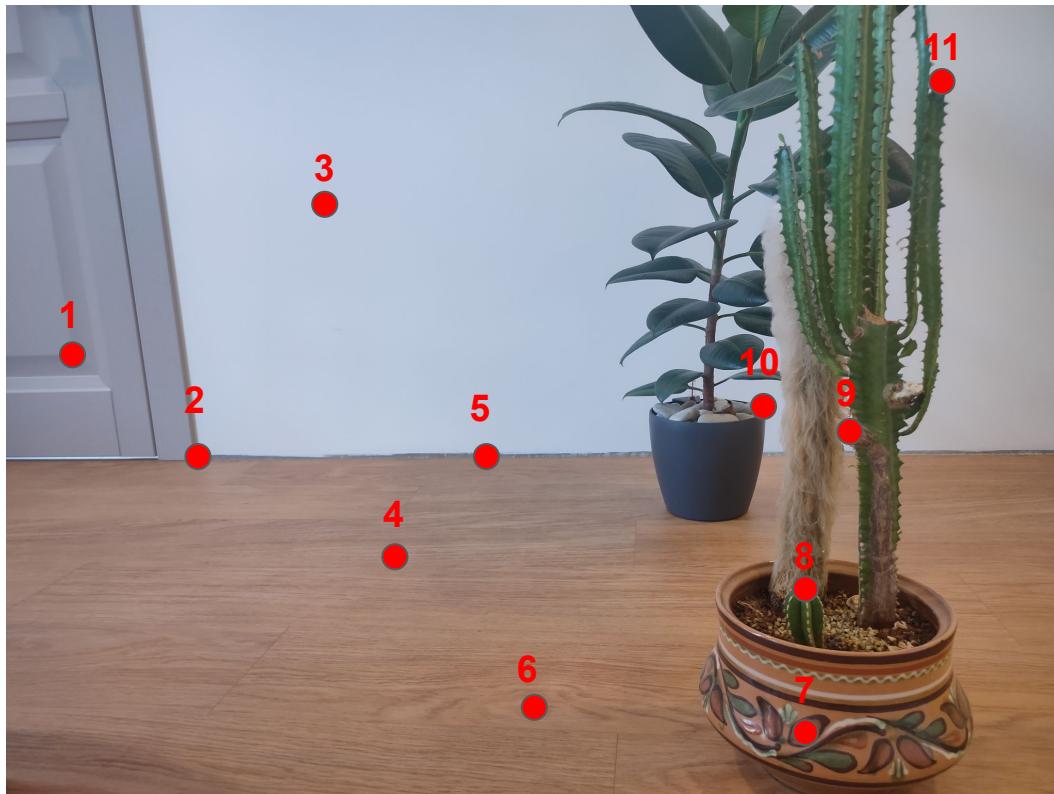
- **Detector:** Инвариантно к искажениям выбрать ключевые точки
- **Descriptor:** Построить для каждой точки описание в виде вектора чисел

Статья: [Distinctive Image Features from Scale-Invariant Keypoints, G. Lowe, 2004](#)

Key Points: Detector - какие точки нам подходят?



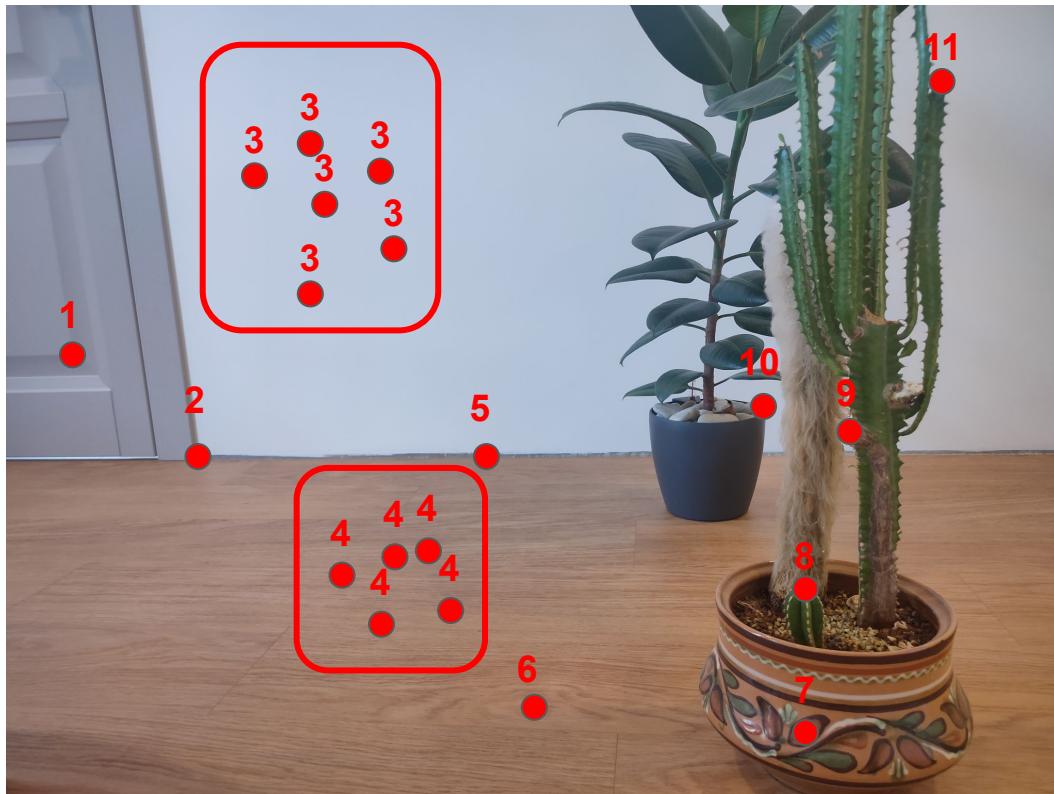
Key Points: Detector - какие точки нам подходят?



Инвариантность: фотография с другого ракурса должна обнаружить те же точки.

Какие точки однозначно плохие?

Key Points: Detector - какие точки нам подходят?

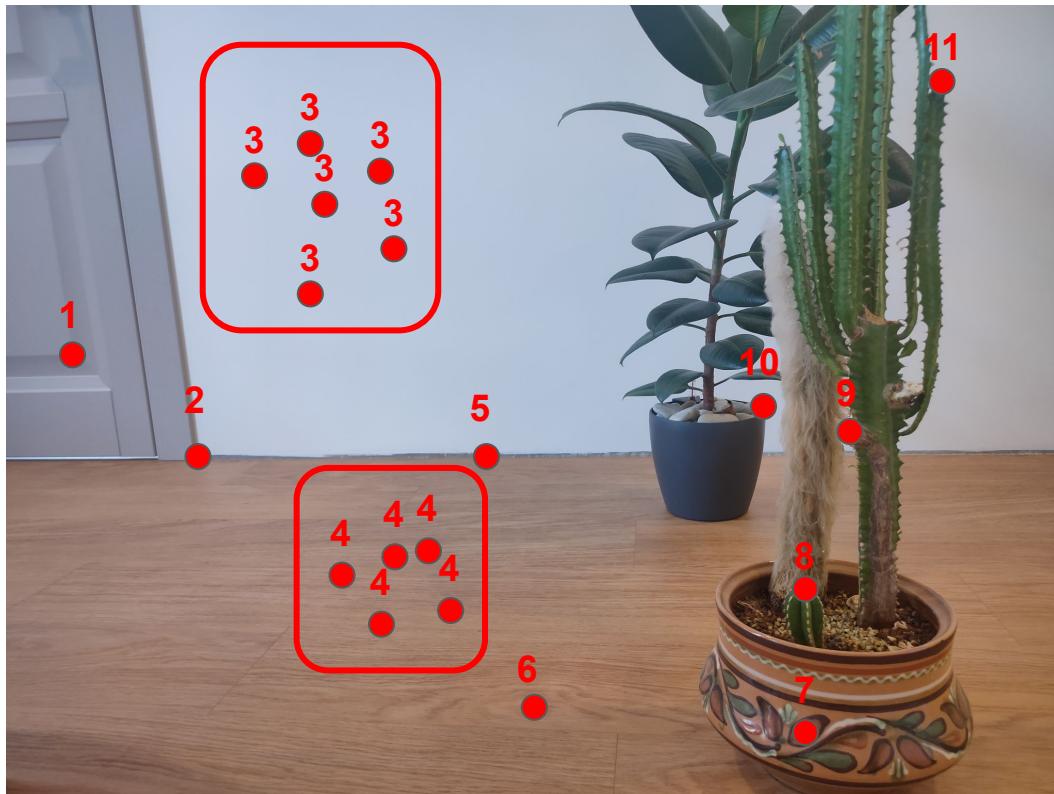


Инвариантность: фотография с другого ракурса должна обнаружить те же точки.

Точка 3: стена белая, зацепиться не за что, текстуры нет, по ней легко “скользить”. Точка не однозначная и большой риск ошибки сопоставления.

Точка 4: зависит, может за текстуру дерева можно однозначно зацепиться, но если кадр размытый - тоже неоднозначность сопоставления.

Key Points: Detector - какие точки нам подходят?



Инвариантность: фотография с другого ракурса должна обнаружить те же точки.

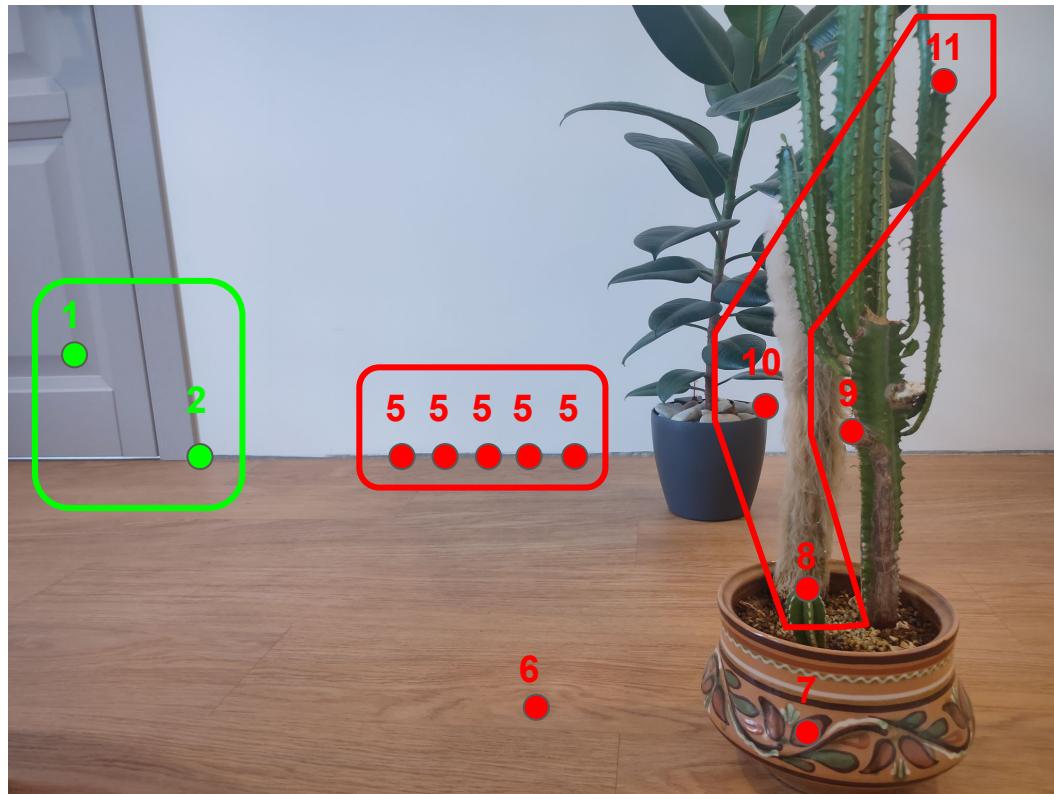
Точка 3: стена белая, зацепиться не за что, текстуры нет, по ней легко “скользить”. Точка не однозначная и большой риск ошибки сопоставления.

Точка 4: зависит, может за текстуру дерева можно однозначно зацепиться, но если кадр размытый - тоже неоднозначность сопоставления.

Какие точки хорошие?

Key Points: Detector - какие точки нам подходят?

Какие точки удобны для сопоставления? Какие к ним требования?



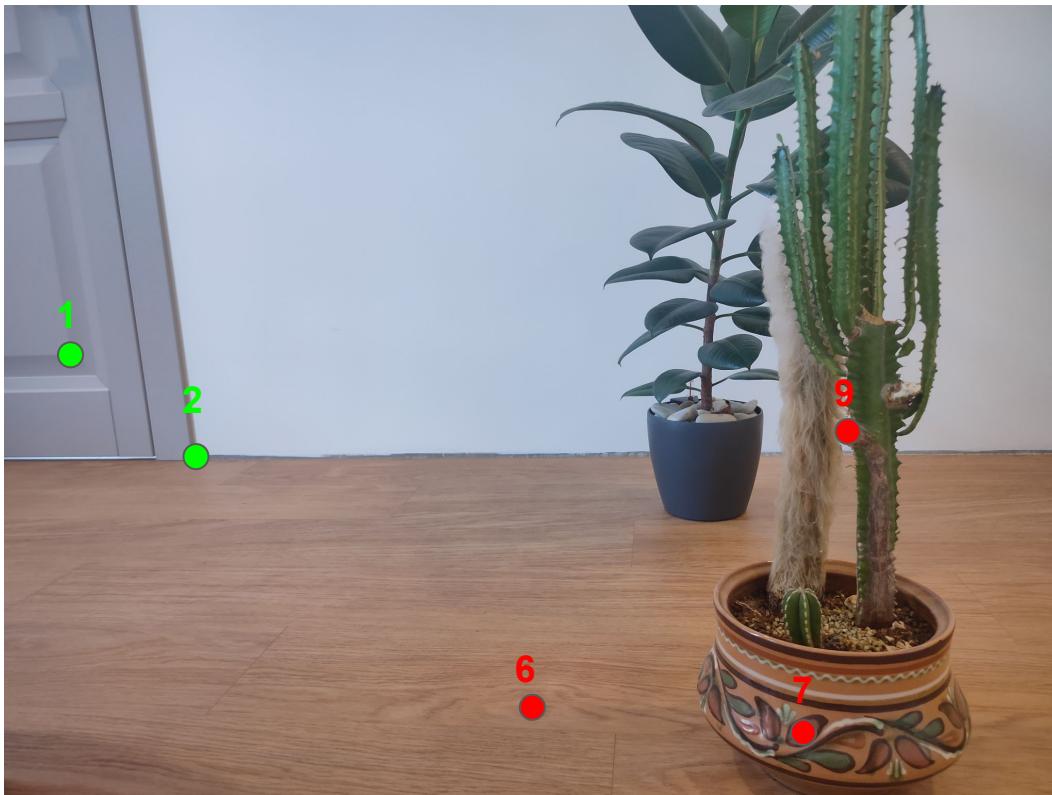
Инвариантность: фотография с другого ракурса должна обнаружить те же точки.

Точка 5: скользит по краю пола.

Точки 1 и 2: хорошие, они **на углу**. Причем **на углу плоскости**. А не на углу чье пространственное положение зависит от положения камеры (см. **точки 8,10,11**), а значит может не воспроизвестись (**точка 10**) или воспроизвестись неточно (**точки 8,11**).

Углы - критерий ключевых точек в **Harris Corner Detector**.

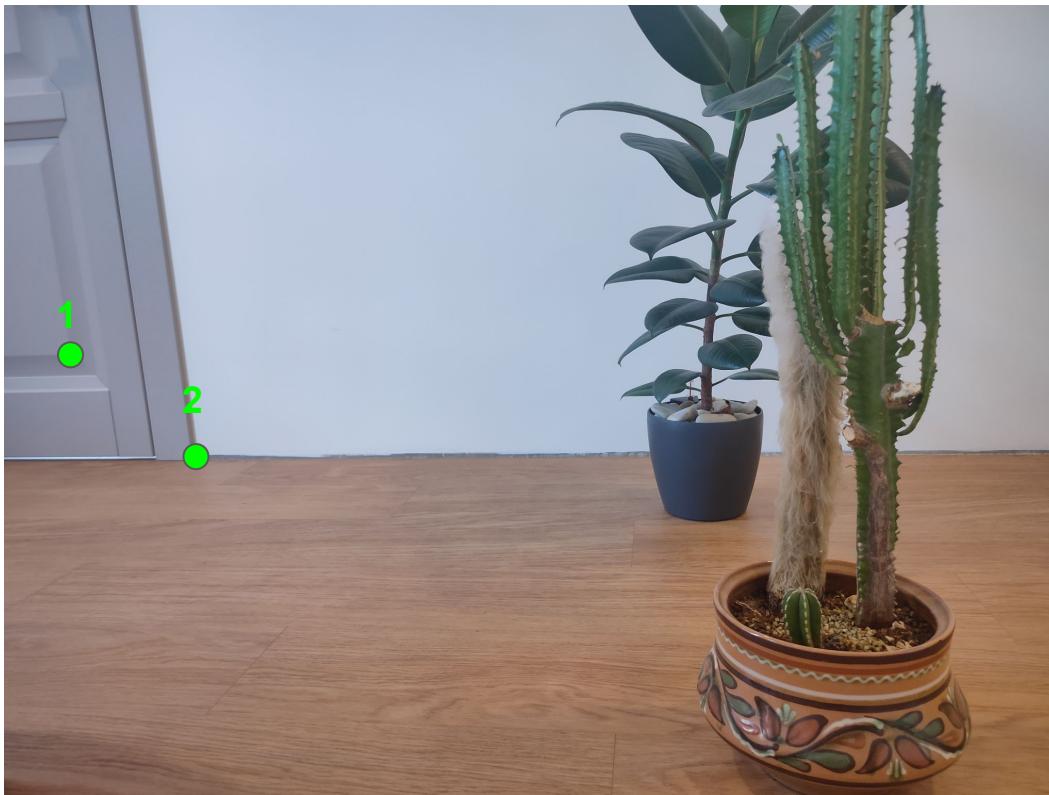
Key Points: Detector - какие точки нам подходят?



Инвариантность: фотография с другого ракурса должна обнаружить те же точки.

Точки 6, 7 и 9: инвариантны или нет?

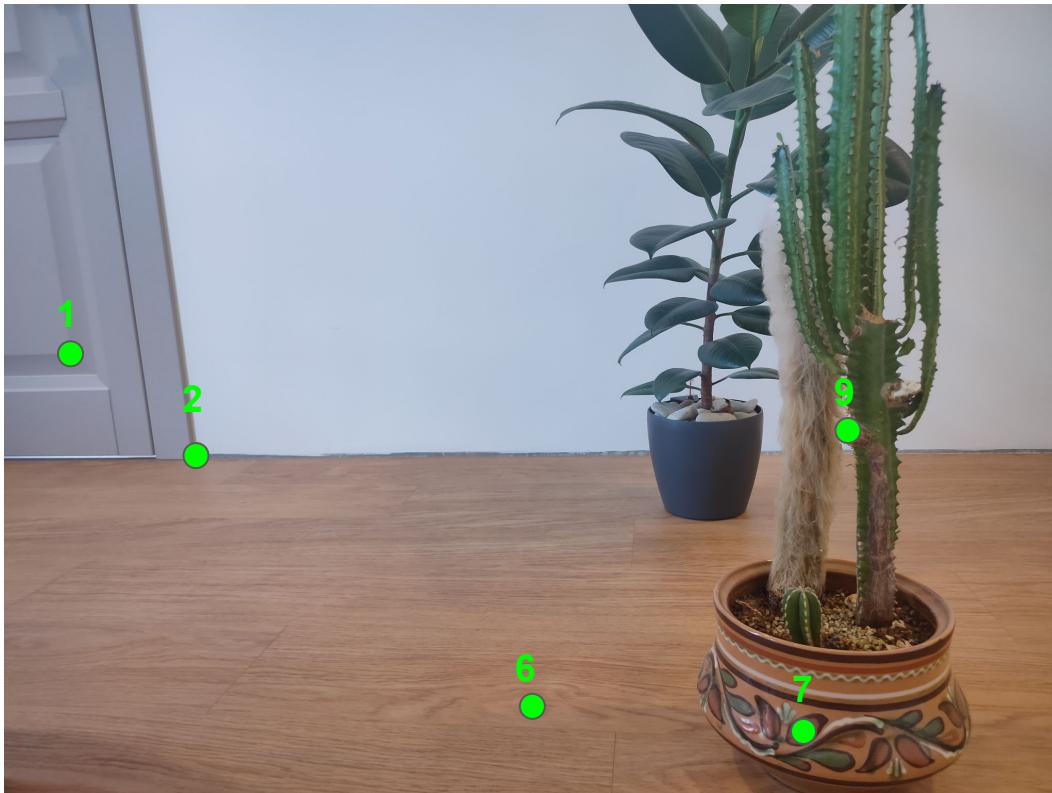
Key Points: Detector - какие точки нам подходят?



Инвариантность: фотография с другого ракурса должна обнаружить те же точки.

Точки 6, 7 и 9: инвариантны или нет?

Key Points: Detector - какие точки нам подходят?



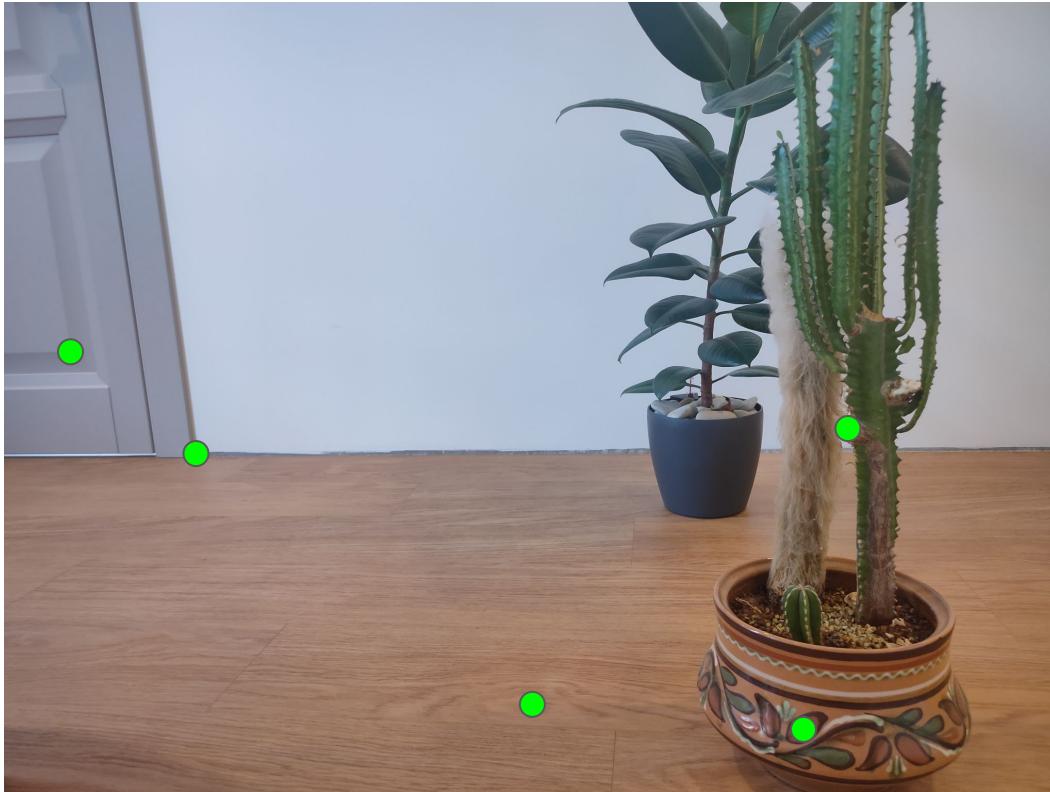
Инвариантность: фотография с другого ракурса должна обнаружить те же точки.

Точки 6, 7 и 9: находятся в центре пятна. Пятно лежащее на плоскости **инвариантно** т.к. его центр можно выбрать точно независимо от ракурса камеры.

Пятна - критерий ключевых точек в **SIFT Detector**.

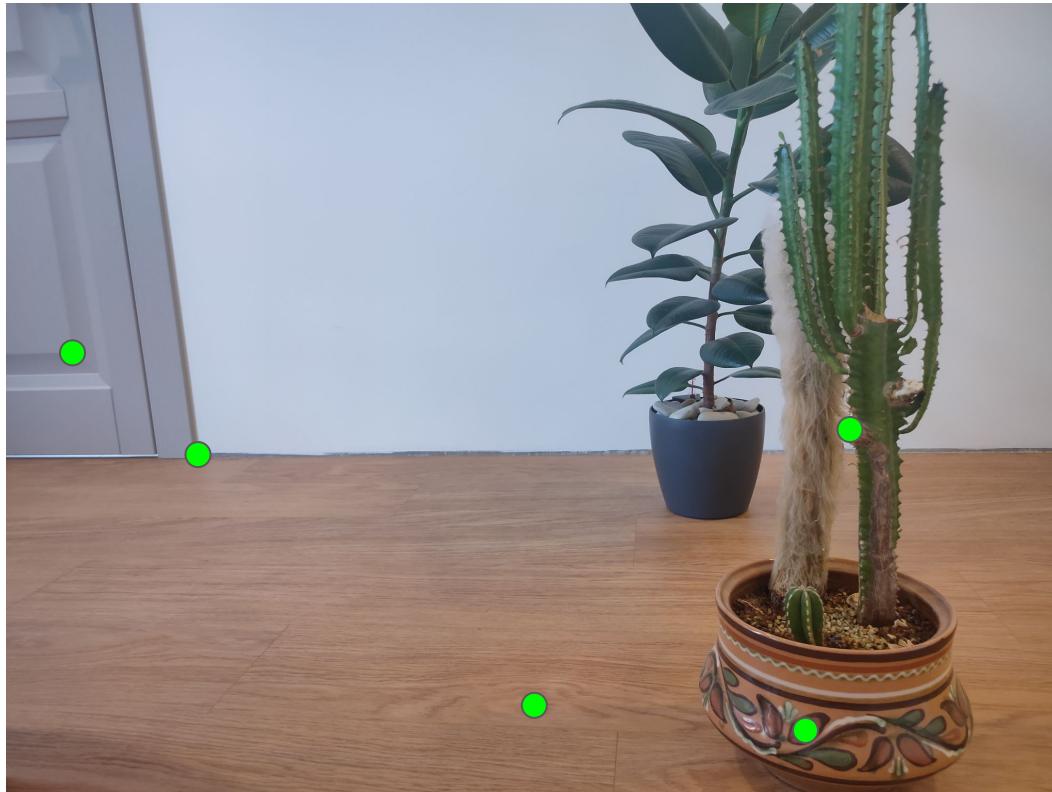
Key Points: Descriptor

Как сопоставлять точки между двумя картинками?



Key Points: Descriptor

Как сопоставлять точки между двумя картинками? Построить им компактное описание - Descriptor.



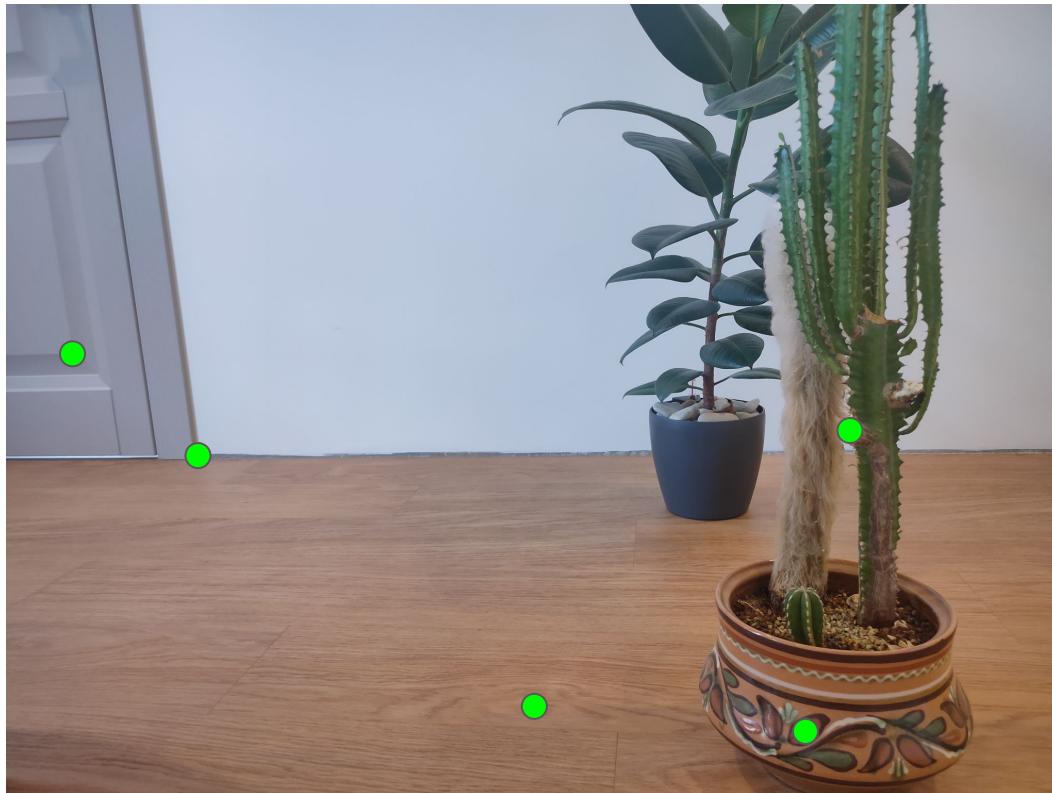
SIFT Descriptor: 128 Float = 512 bytes

Как оценить похожесть двух точек?

Как оценить похожесть двух дескрипторов А и В?

Key Points: Descriptor

Как сопоставлять точки между двумя картинками? Построить им компактное описание - Descriptor.



SIFT Descriptor: 128 Float = 512 bytes

Как оценить похожесть двух точек?

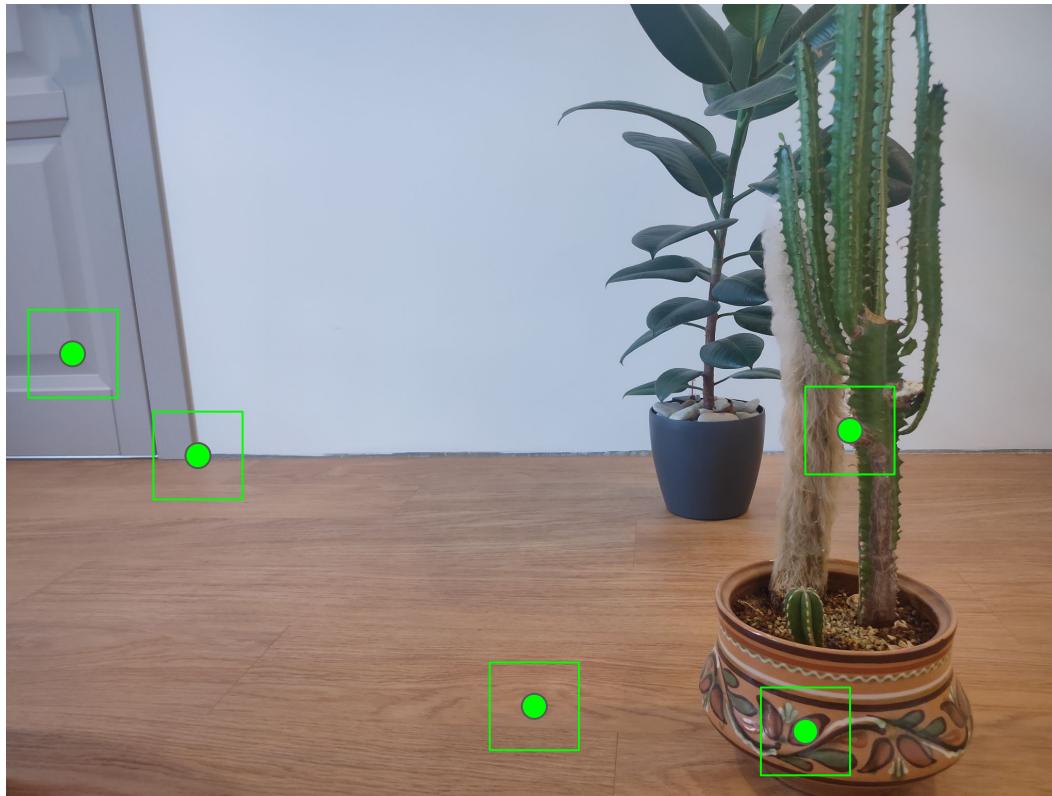
Как оценить похожесть двух дескрипторов A и B?

Посчитать между ними евклидово расстояние:

$$dist(A, B) = \sqrt{\sum_{i=1}^{128} (A_i - B_i)^2}$$

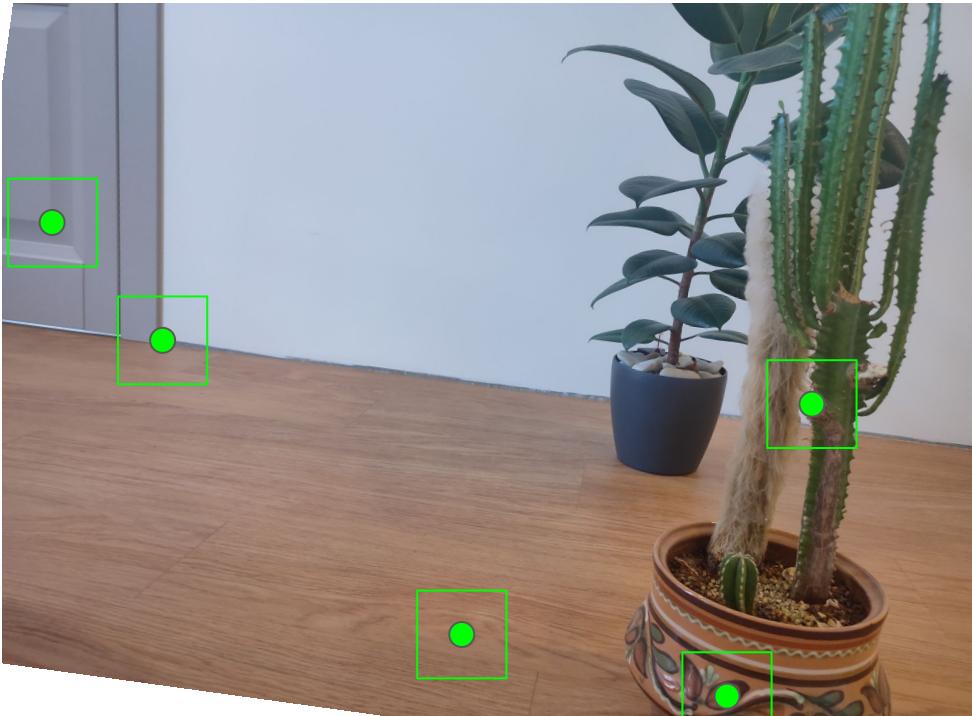
Key Points: Descriptor

Какие есть проблемы с **инвариантностью Descriptor-а?**



Key Points: Descriptor

Какие есть проблемы с **инвариантностью Descriptor-а?**

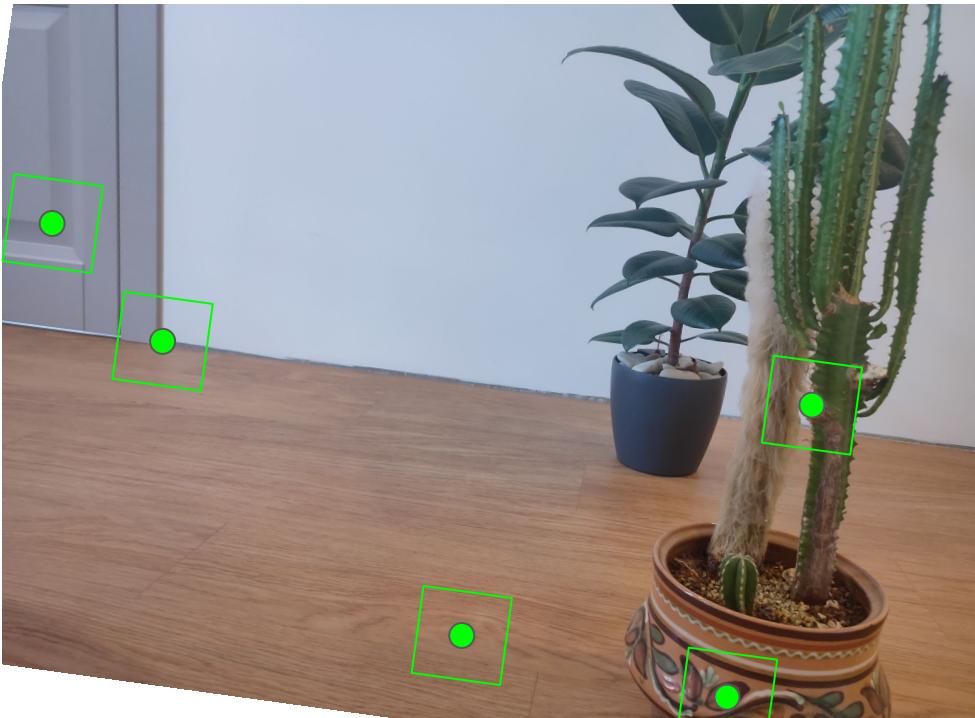


Если другая фотография под углом -
информация в локальной окрестности
(патч) одних и тех же ключевых точек
сильно меняется.

Т.е. просто брать квадрат вокруг точки =
неинвариантный к повороту
дескриптор.

Key Points: Descriptor

Какие есть проблемы с **инвариантностью Descriptor-а?**



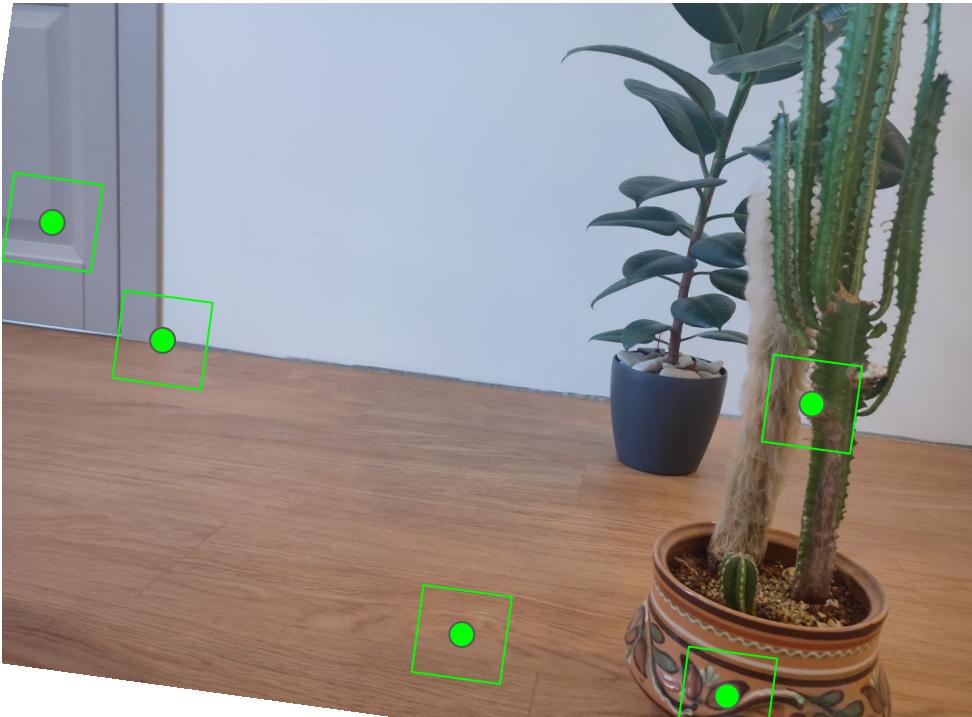
Если другая фотография под углом -
информация в локальной окрестности
(патч) одних и тех же ключевых точек
сильно меняется.

Т.е. просто брать квадрат вокруг точки =
неинвариантный к повороту
дескриптор.

Значит было бы хорошо повернуть **патчи**
чтобы они были под тем же углом что и в
оригинальной картинке. Но мы не знаем
какой угол поворота!

Key Points: Descriptor

Какие есть проблемы с **инвариантностью Descriptor-а?**



Если другая фотография под углом -
информация в локальной окрестности
(патч) одних и тех же ключевых точек
сильно меняется.

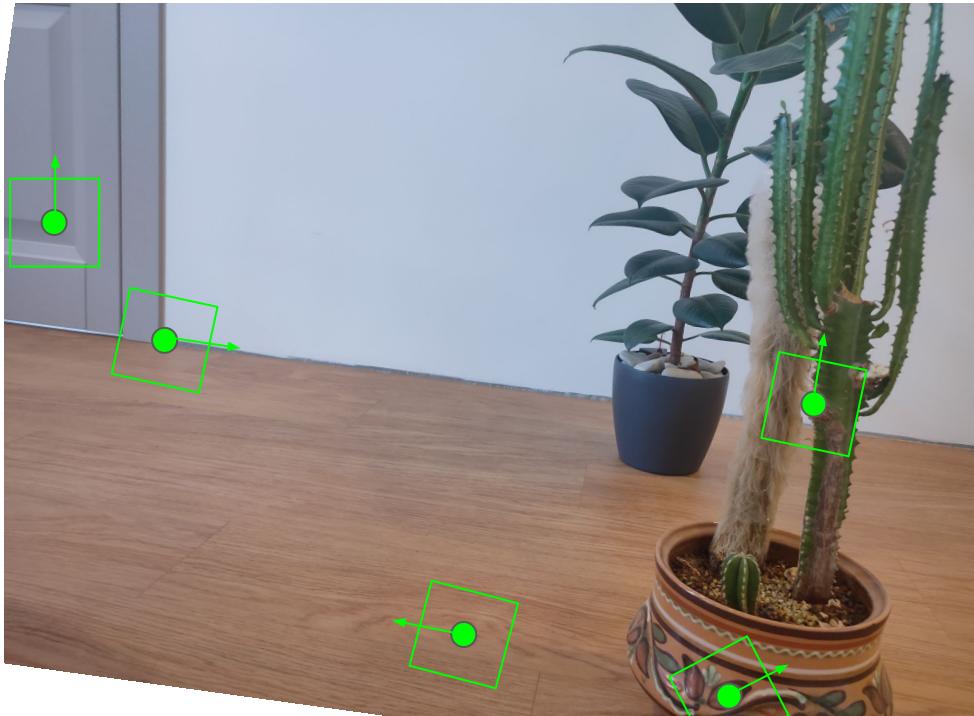
Т.е. просто брать квадрат вокруг точки =
неинвариантный к повороту
дескриптор.

Значит было бы хорошо повернуть **патчи**
чтобы они были под тем же углом что и в
оригинальной картинке. Но мы не знаем
какой угол поворота!

+ Пришлось бы **пересчитывать**
дескрипторы точек фотографии много раз
- для каждого сопоставления
(т.к. могут быть разные повороты).

Key Points: Descriptor

Какие есть проблемы с **инвариантностью Descriptor-а?**

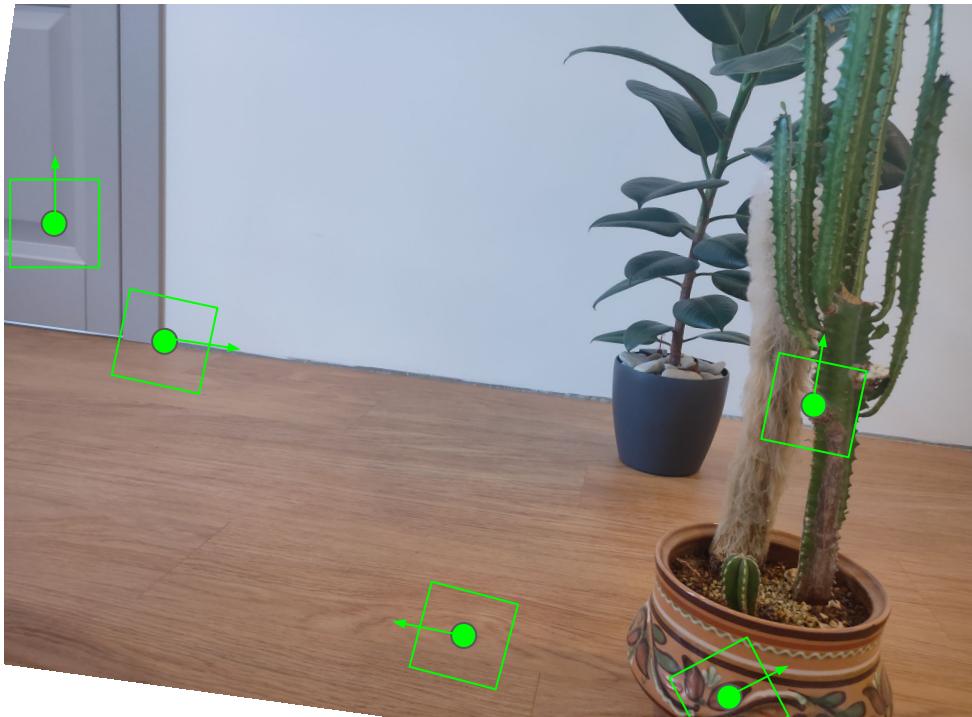


- 1) **Инвариантность по повороту:** ключевая точка выбирает направление (например самое популярное направление градиента в локальной окрестности).

Локальный патч повернут соответственно.

Key Points: Descriptor

Какие есть проблемы с **инвариантностью Descriptor-a?**



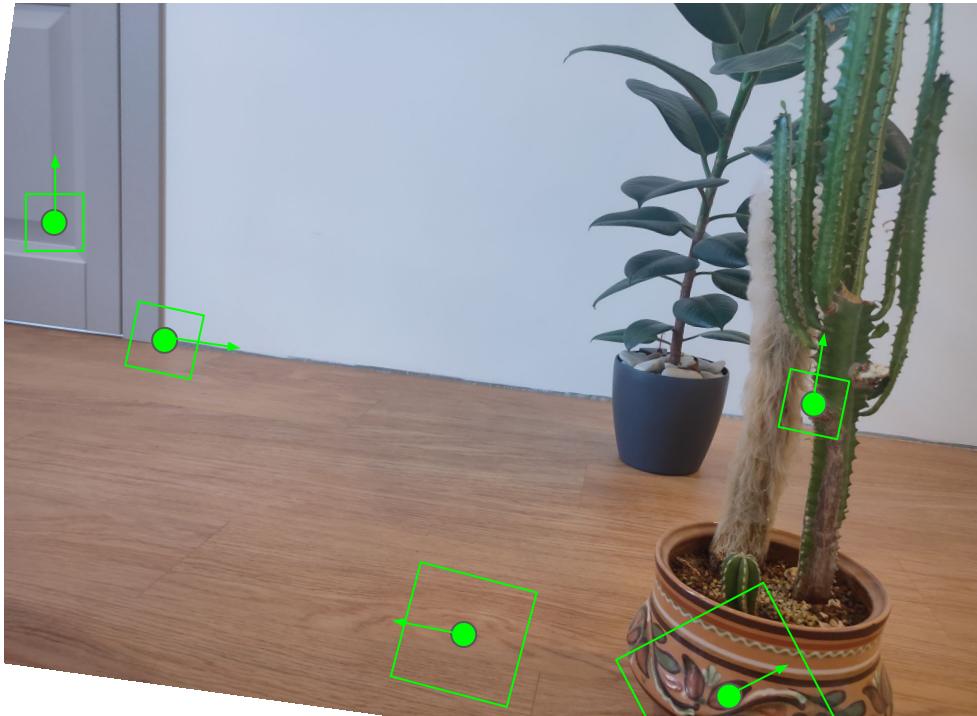
- 1) **Инвариантность по повороту:** ключевая точка выбирает направление (например самое популярное направление градиента в локальной окрестности).

Локальный патч повернут соответственно.

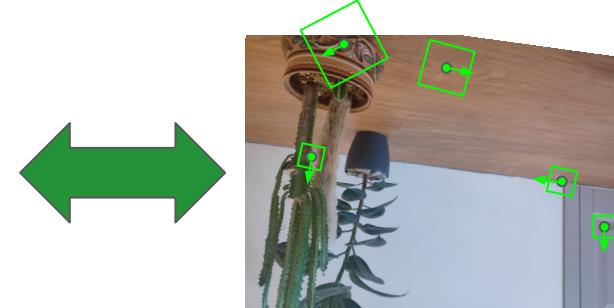
Какая еще инвариантность нужна?

Key Points: Descriptor

Какие есть проблемы с **инвариантностью Descriptor-a?**



- 1) Инвариантность по повороту
- 2) Инвариантность по масштабу

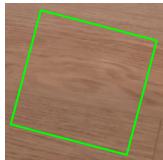


Key Points: Descriptor

Какие есть проблемы с **инвариантностью Descriptor-а?**

- 1) Инвариантность по повороту
- 2) Инвариантность по масштабу
- 3) Инвариантность по описанию

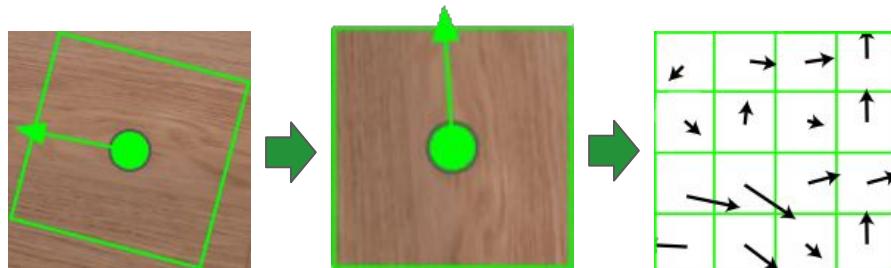
Теперь надо придумать алгоритм который представляет в виде 128 вещественных чисел локальный патч (поворнутый и “правильного” размера).



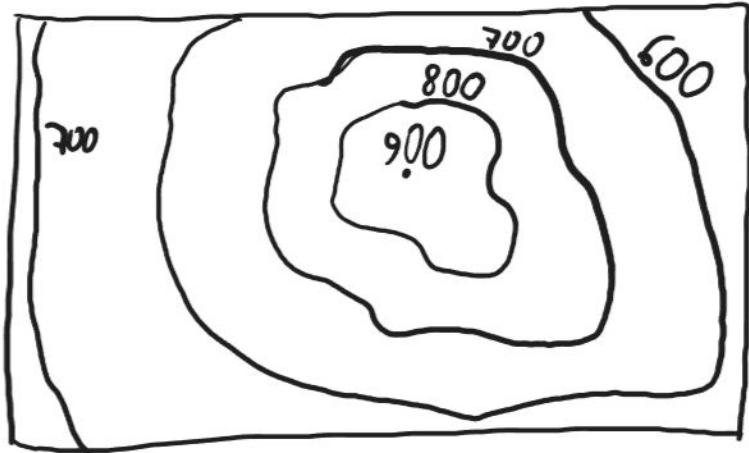
Key Points: Descriptor

Какие есть проблемы с **инвариантностью Descriptor-a?**

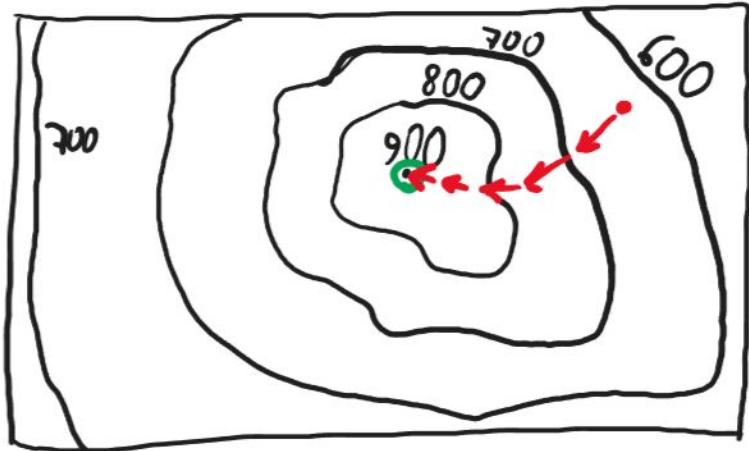
- 1) Инвариантность по повороту
- 2) Инвариантность по масштабу
- 3) Инвариантность по описанию



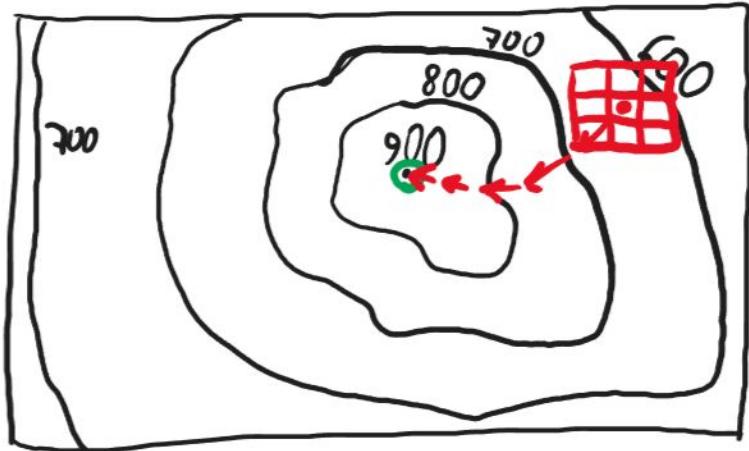
Вычисление градиента в картинке оператором Собеля



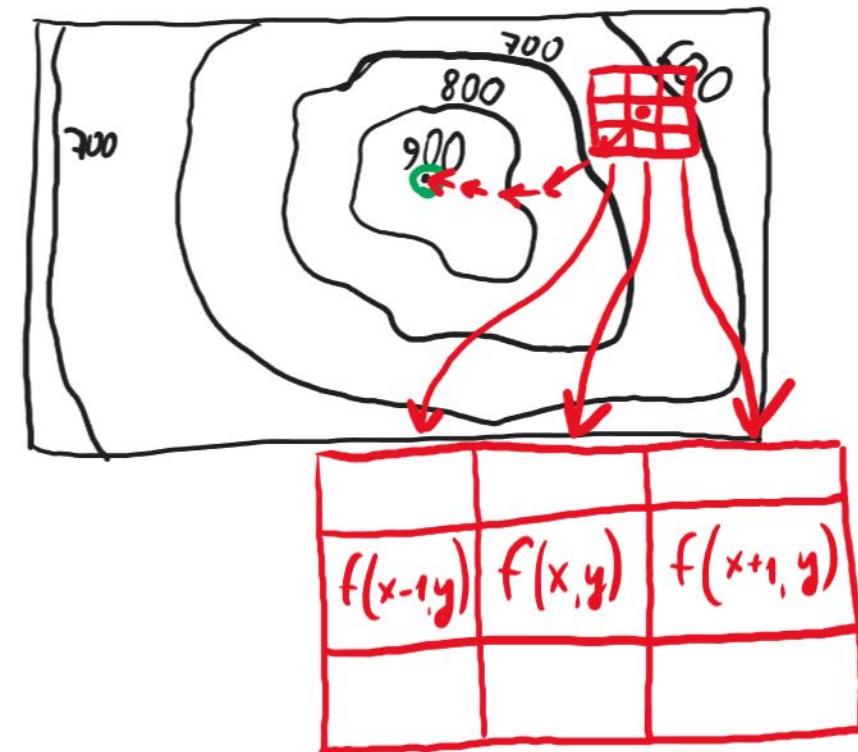
Вычисление градиента в картинке оператором Собеля



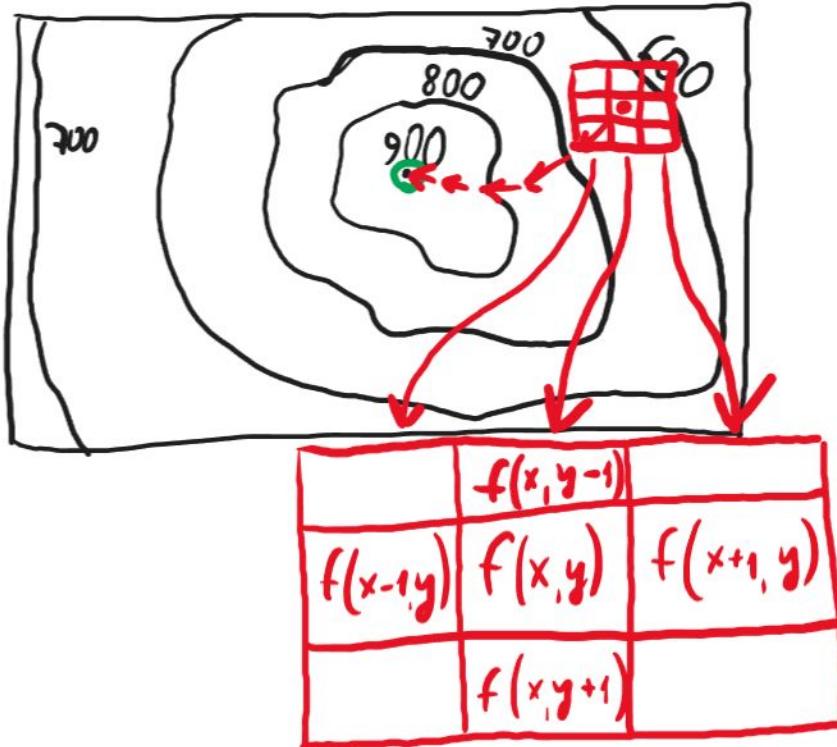
Вычисление градиента в картинке оператором Собеля



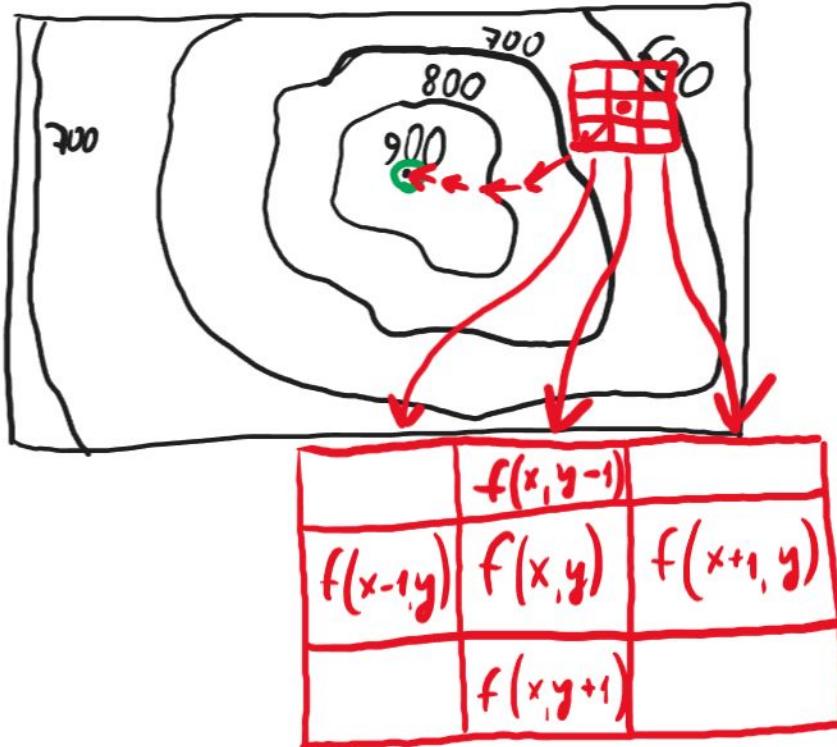
Вычисление градиента в картинке оператором Собеля



Вычисление градиента в картинке оператором Собеля

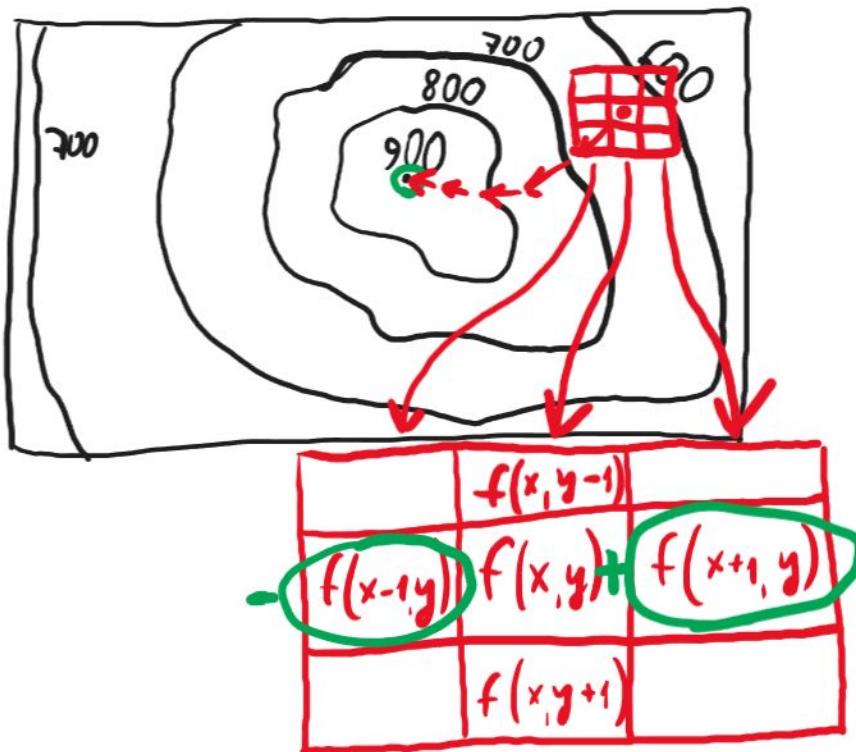


Вычисление градиента в картинке оператором Собеля



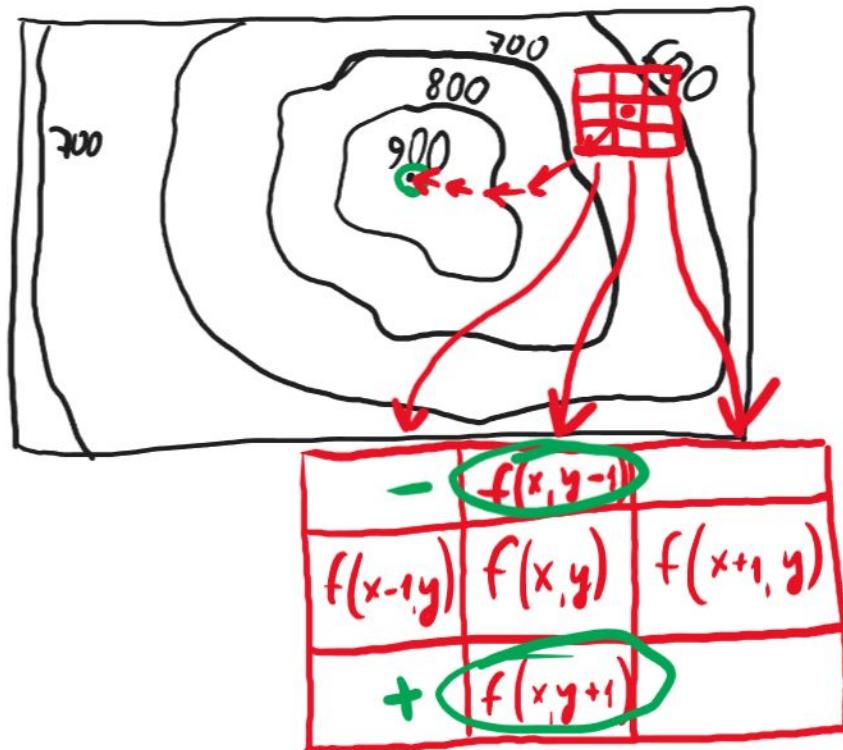
$$f'_x(x, y) = ?$$

Вычисление градиента в картинке оператором Собеля



$$f'_x(x, y) = f(x+1, y) - f(x-1, y)$$

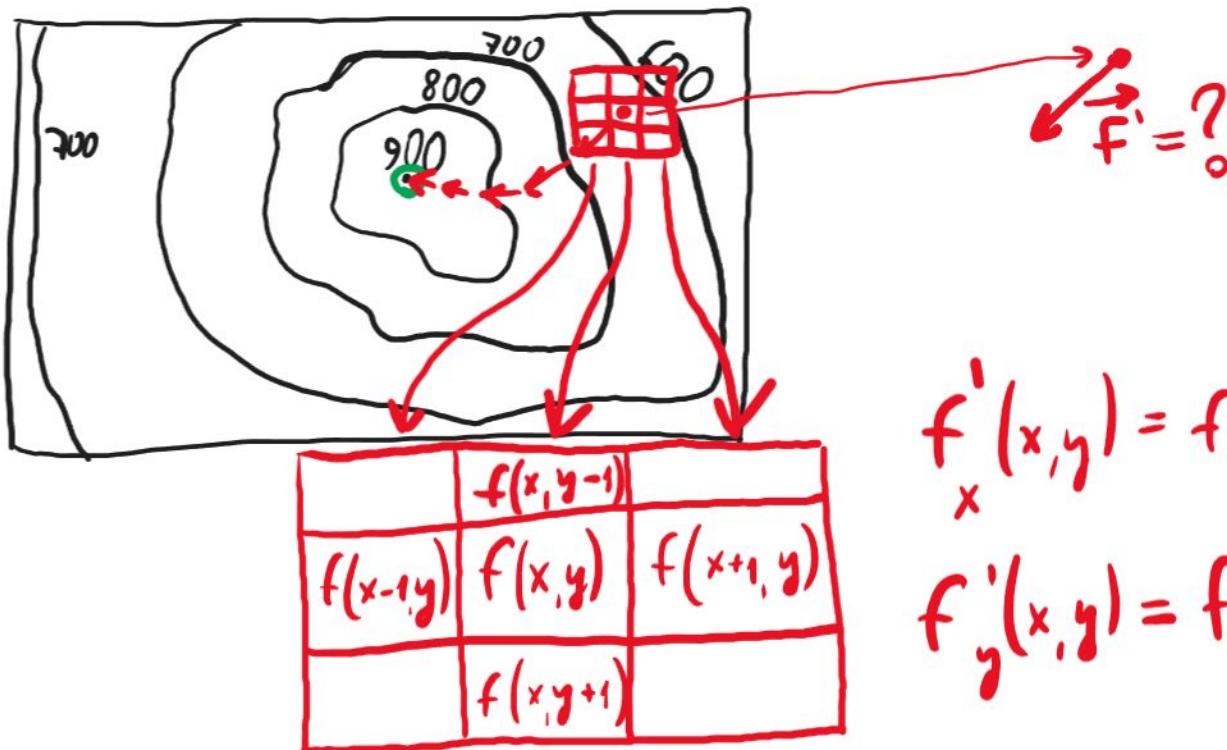
Вычисление градиента в картинке оператором Собеля



$$f'_x(x, y) = f(x+1, y) - f(x-1, y)$$

$$f'_y(x, y) = f(x, y+1) - f(x, y-1)$$

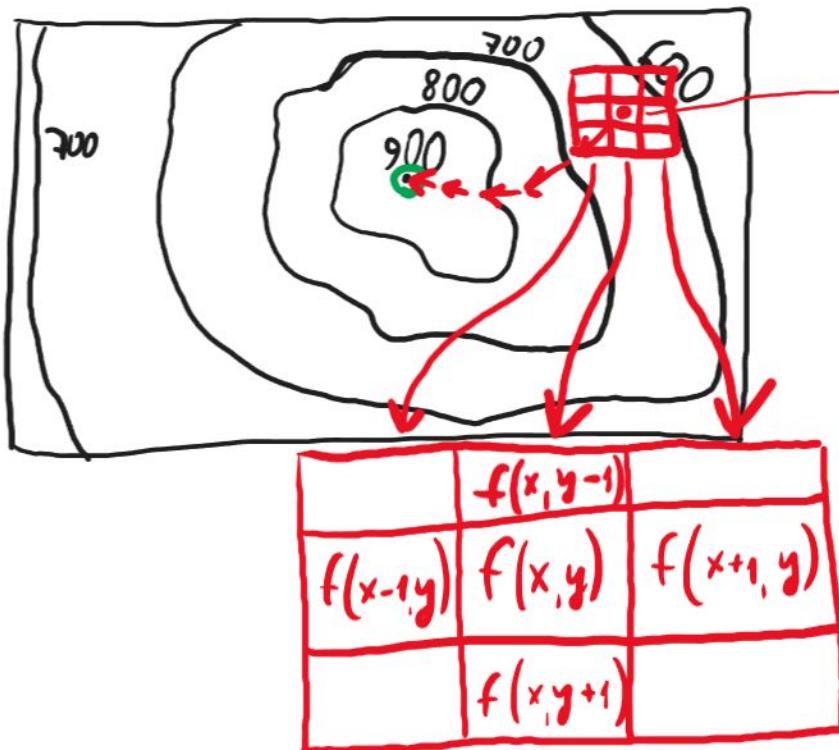
Вычисление градиента в картинке оператором Собеля



$$f'_x(x, y) = f(x+1, y) - f(x-1, y)$$

$$f'_y(x, y) = f(x, y+1) - f(x, y-1)$$

Вычисление градиента в картинке оператором Собеля

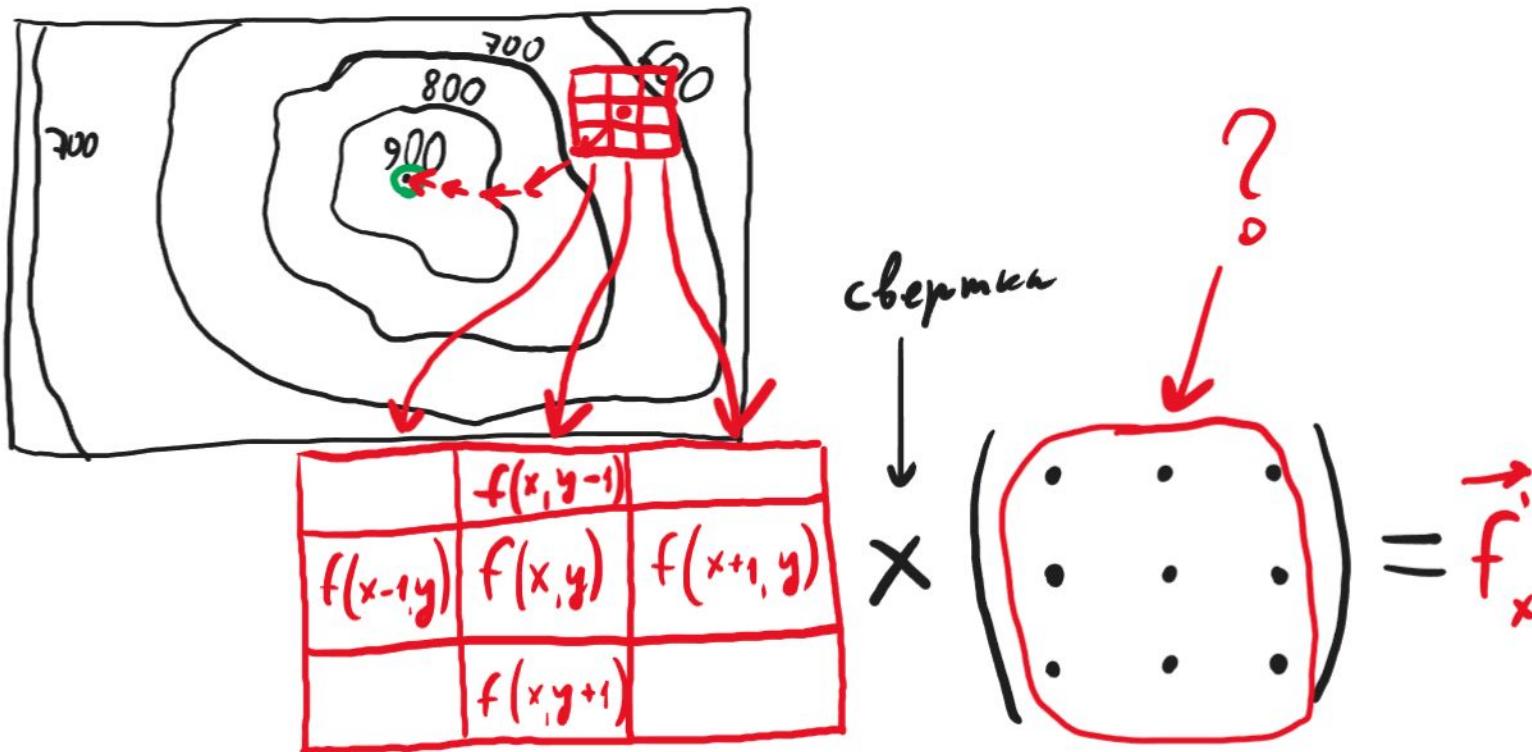


$$\vec{f}' = \underline{(f'_x(x, y); f'_y(x, y))}$$

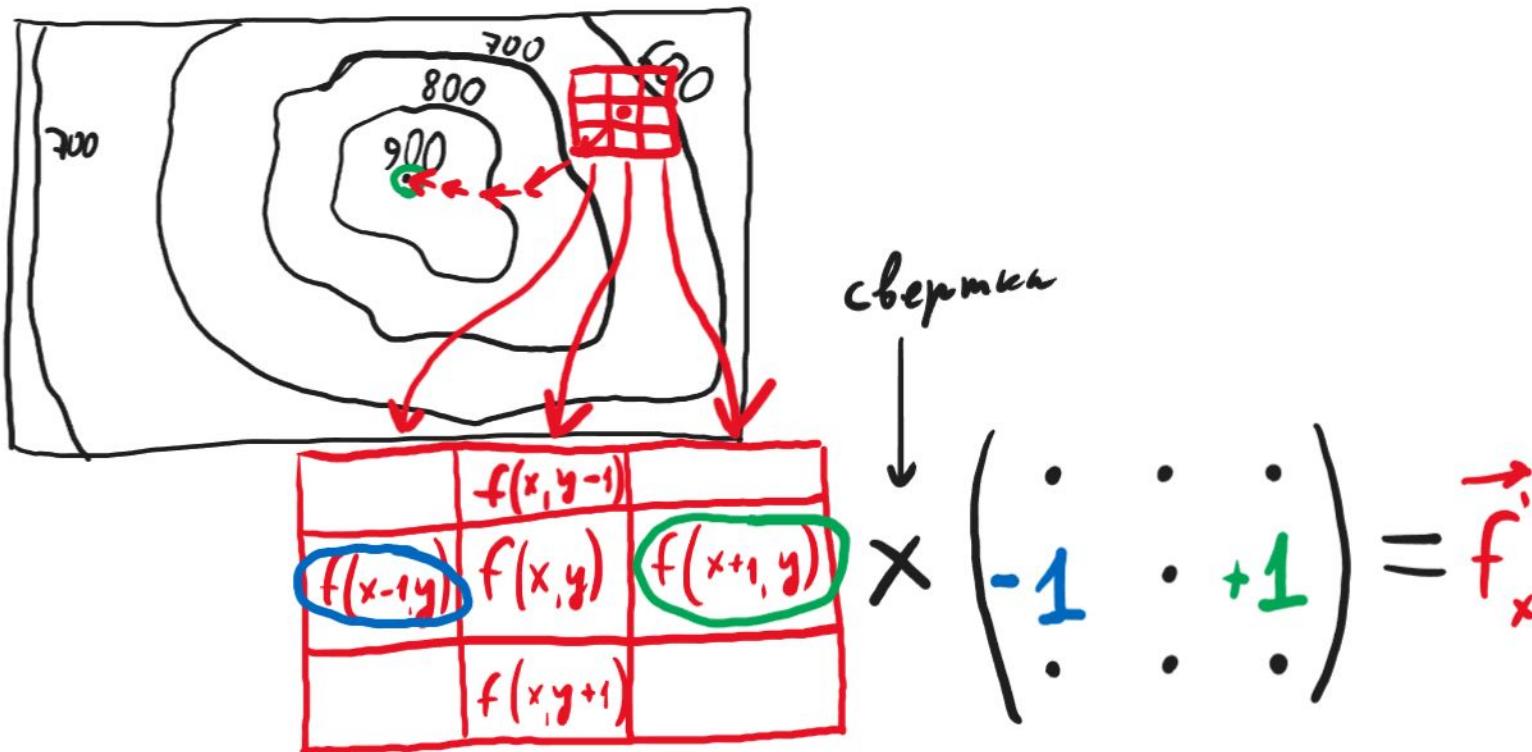
$$f'_x(x, y) = f(x+1, y) - f(x-1, y)$$

$$f'_y(x, y) = f(x, y+1) - f(x, y-1)$$

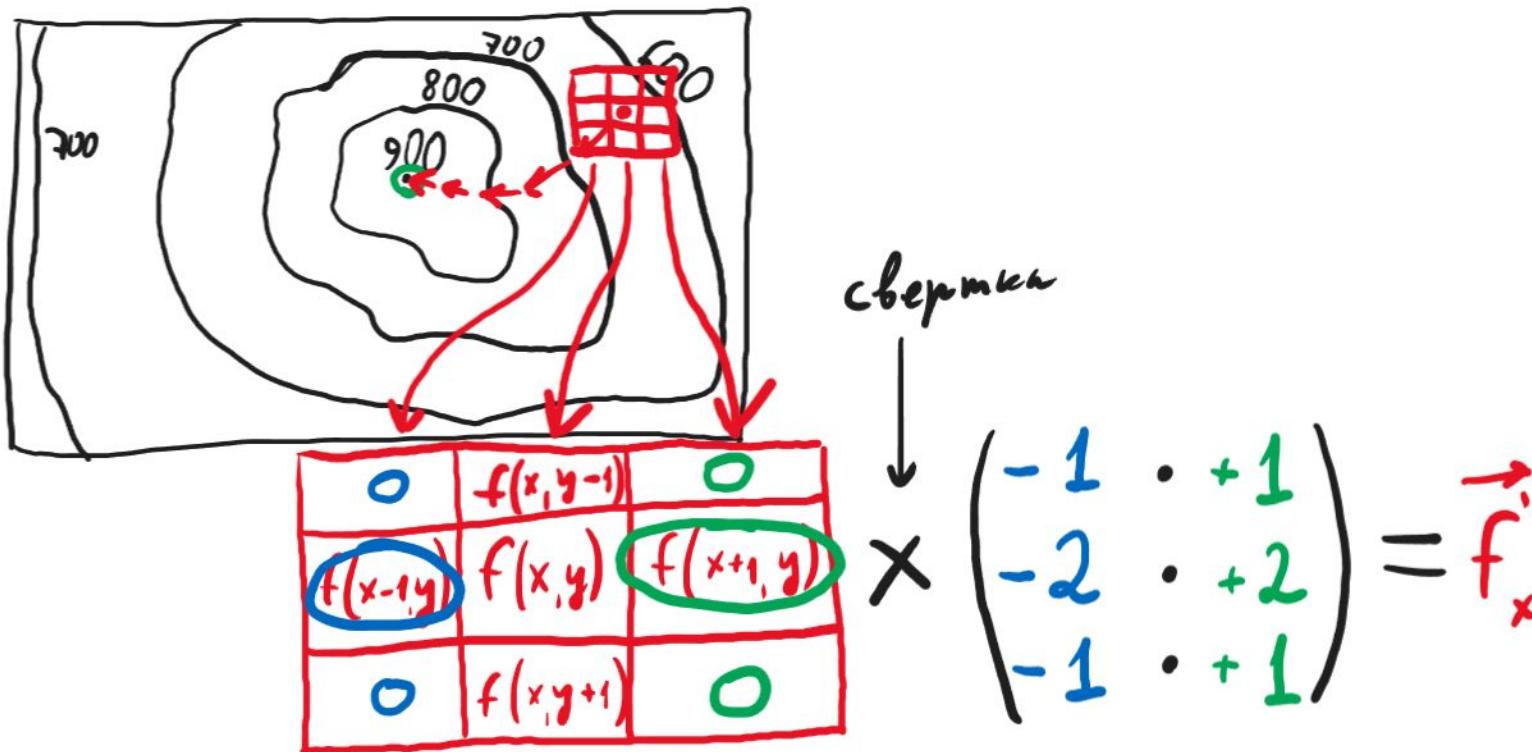
Вычисление градиента в картинке оператором Собеля



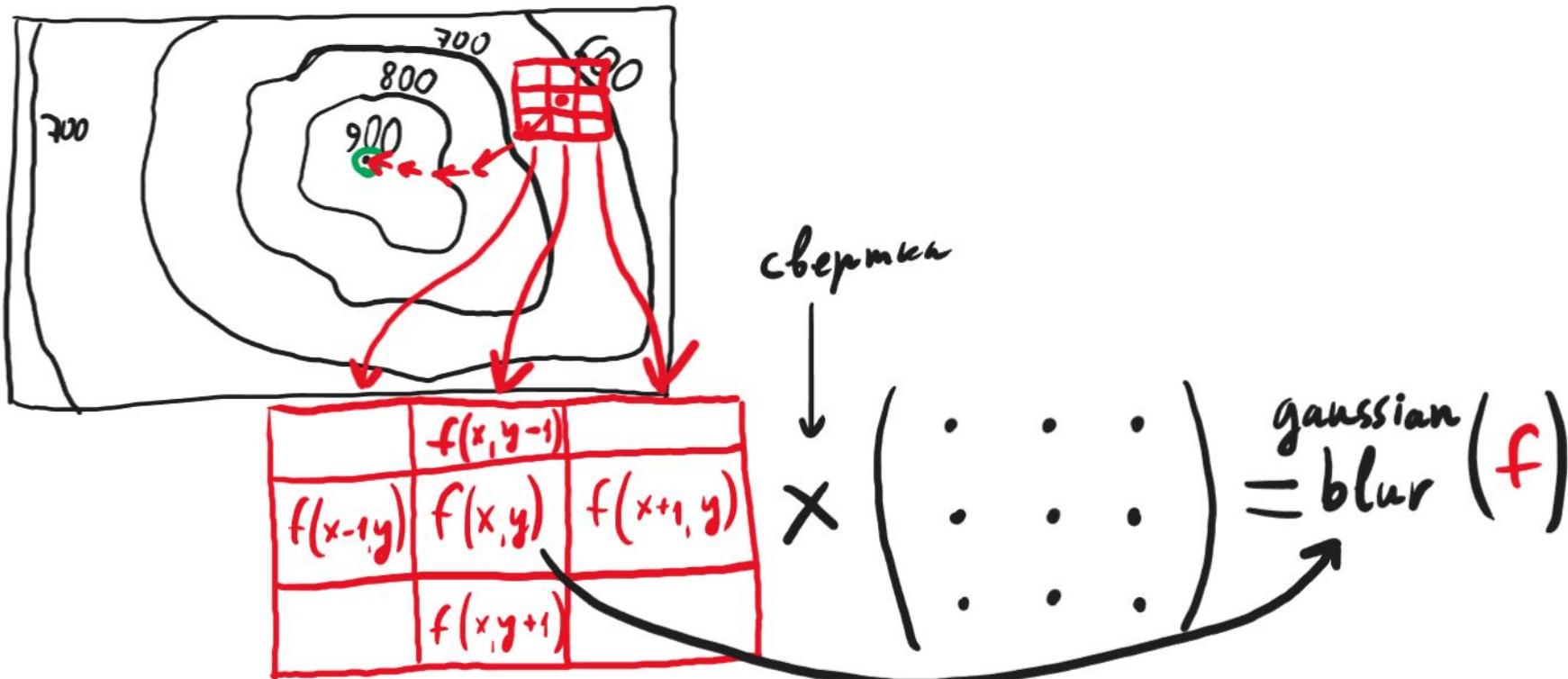
Вычисление градиента в картинке оператором Собеля



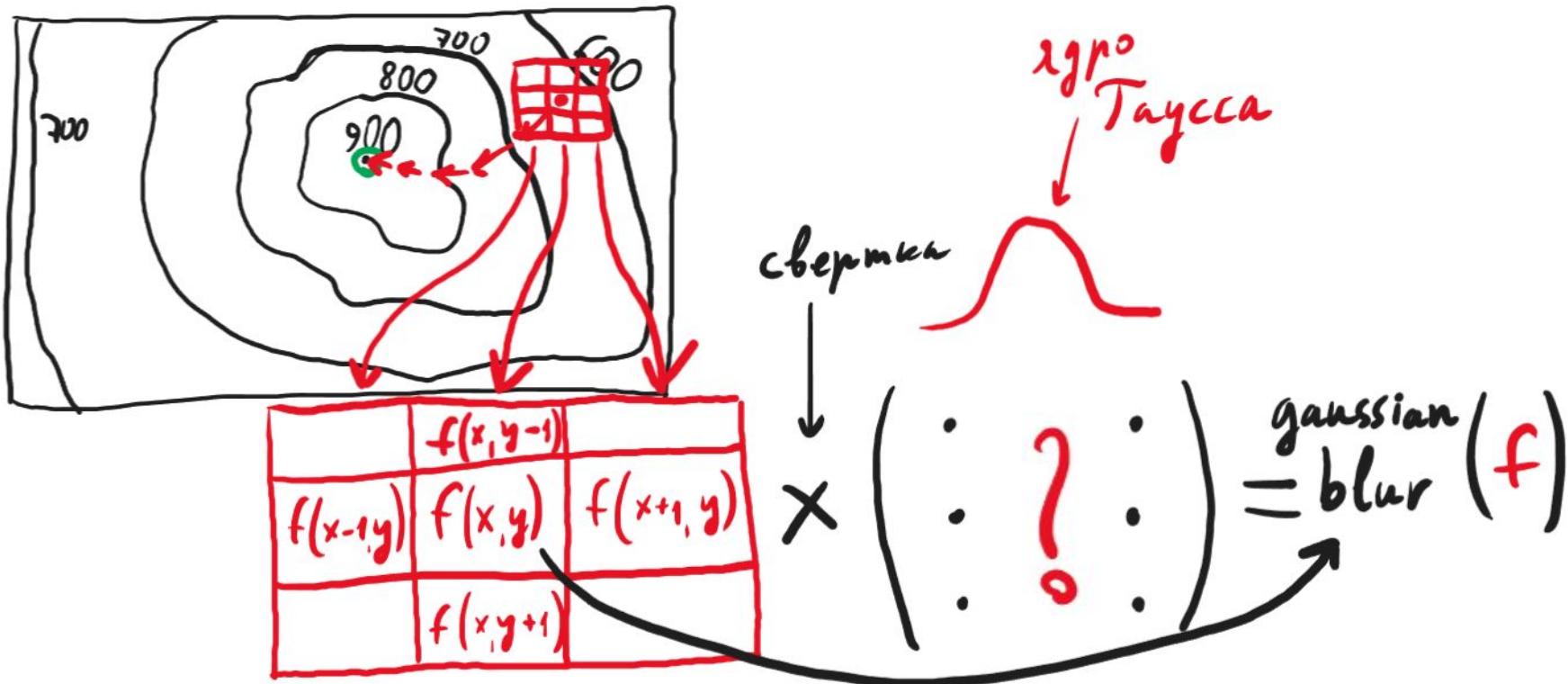
Вычисление градиента в картинке оператором Собеля



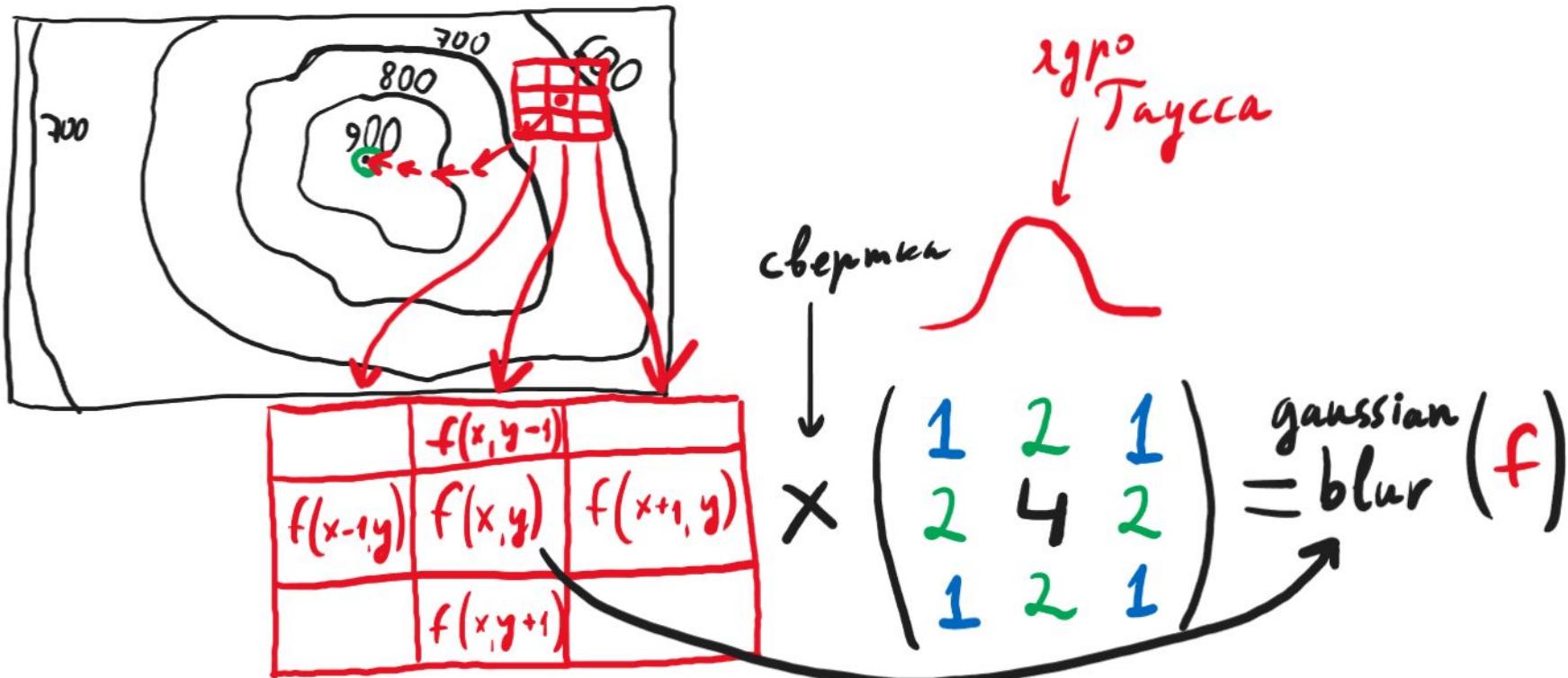
Размытие картинки сверткой с ядром Гаусса



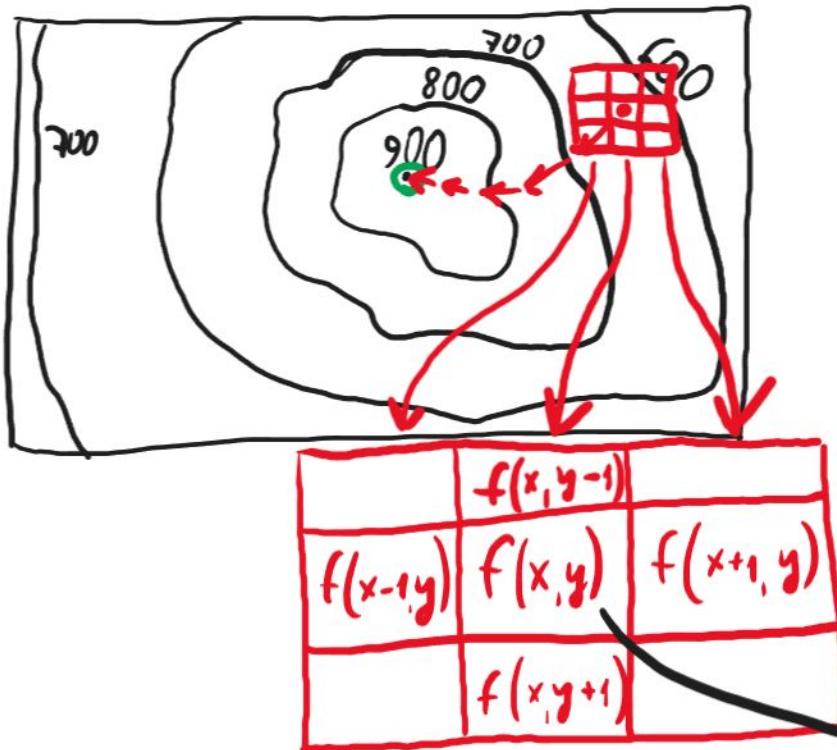
Размытие картинки сверткой с ядром Гаусса



Размытие картинки сверткой с ядром Гаусса



Размытие картинки сверткой с ядром Гаусса



свертка

\downarrow

\times

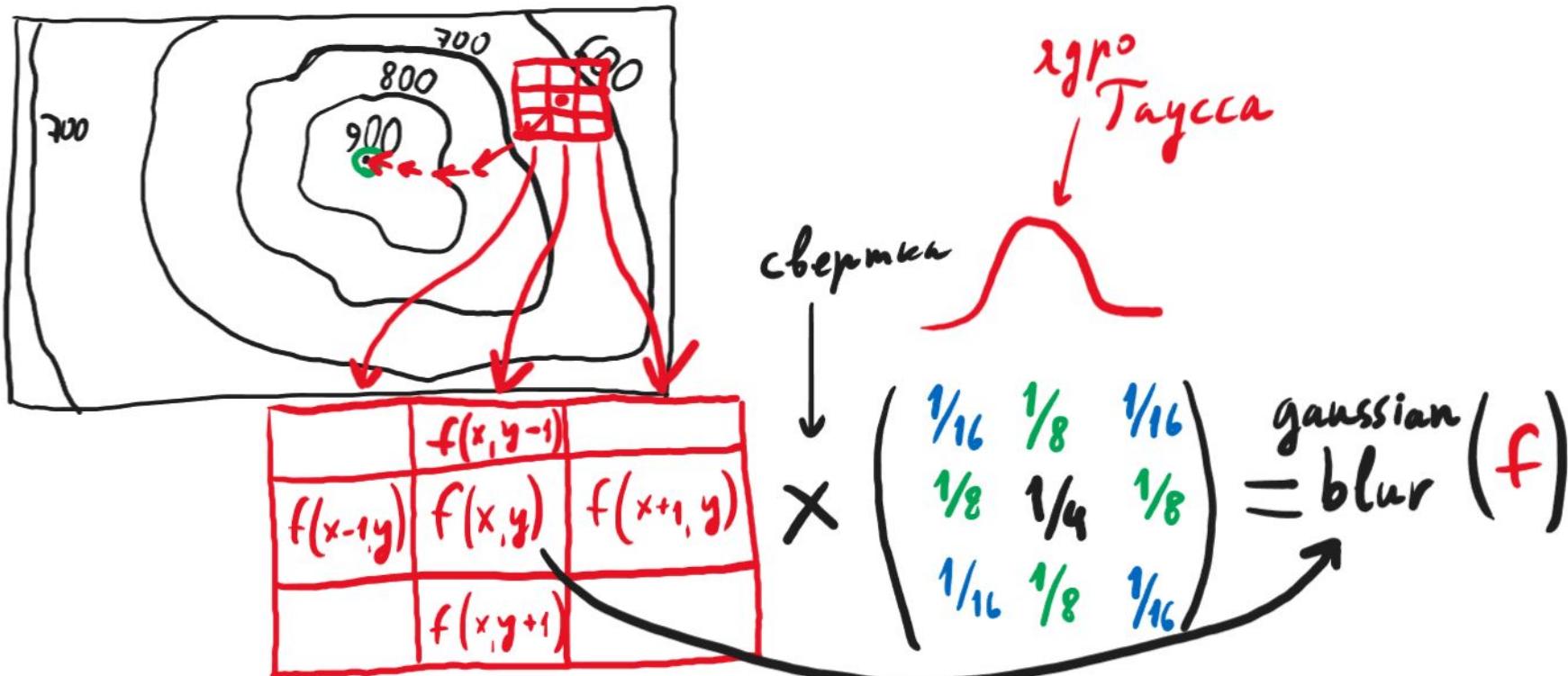
$\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$

$\xrightarrow{\text{ядро Гаусса}}$

$= \text{gaussian blur } (f)$

A yellow illustration of a thinking person is shown on the right, with a red arrow pointing from the text "ядро Гаусса" to their head.

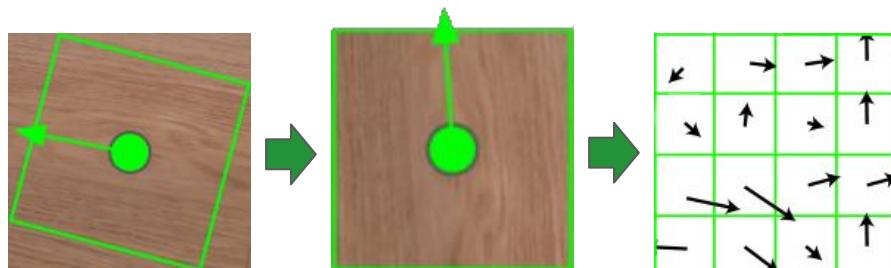
Размытие картинки сверткой с ядром Гаусса



Key Points: Descriptor

Какие есть проблемы с **инвариантностью Descriptor-a?**

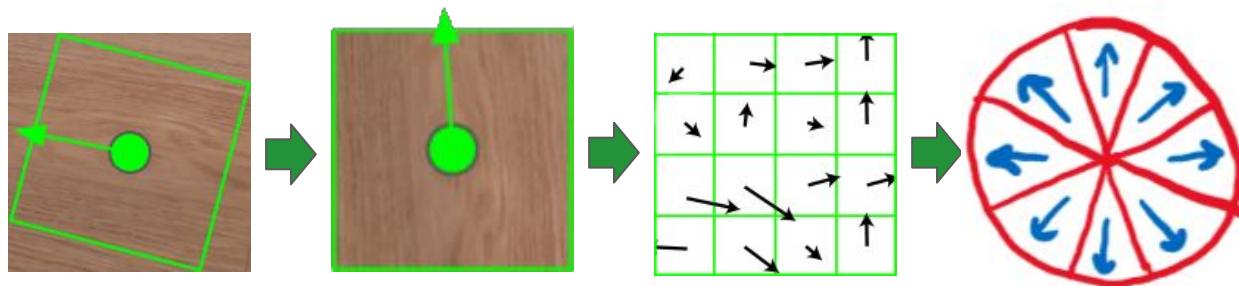
- 1) Инвариантность по повороту
- 2) Инвариантность по масштабу
- 3) Инвариантность по описанию



Key Points: Descriptor

Какие есть проблемы с **инвариантностью Descriptor-a?**

- 1) Инвариантность по повороту
- 2) Инвариантность по масштабу
- 3) Инвариантность по описанию



т.е. все градиенты голосуют за 8 направлений
т.е. 8 корзин-направлений в гистограмме

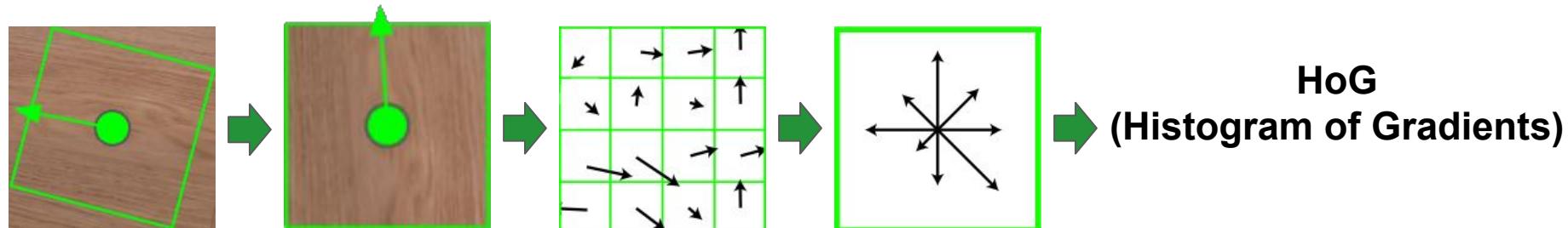
Key Points: Descriptor

Какие есть проблемы с **инвариантностью Descriptor-a?**

HoG = N чисел: одно число - одно направление.

Число получается суммой длин градиентов которые указывают в этом направлении.

- 1) Инвариантность по повороту
- 2) Инвариантность по масштабу
- 3) Инвариантность по описанию



т.е. все градиенты голосуют за 8 направлений
т.е. 8 корзин-направлений в гистограмме

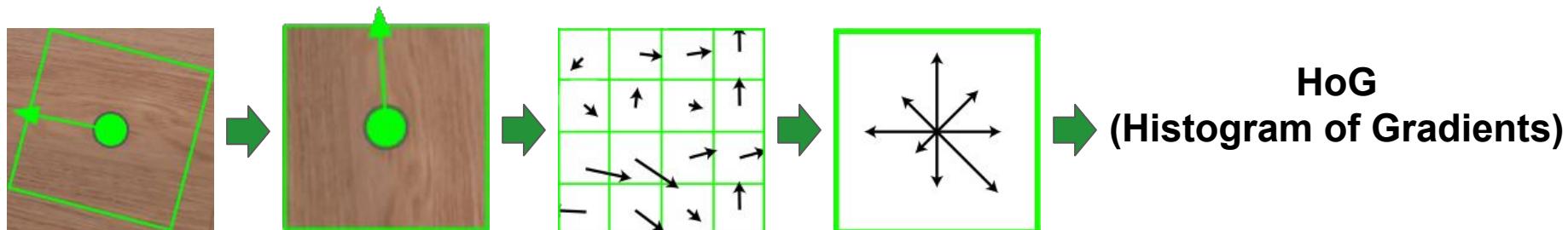
Key Points: Descriptor

Какие есть проблемы с **инвариантностью Descriptor-a?**

HoG = N чисел: одно число - одно направление.

Число получается суммой длин градиентов которые указывают в этом направлении.

- 1) Инвариантность по повороту
- 2) Инвариантность по масштабу
- 3) Инвариантность по описанию:
Какие изменения картинки почти не меняют HoG?



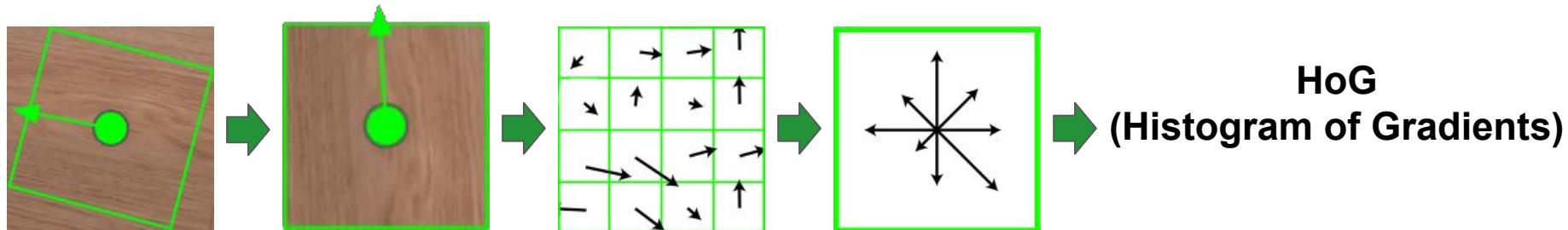
Key Points: Descriptor

Какие есть проблемы с **инвариантностью Descriptor-a?**

HoG = N чисел: одно число - одно направление.

Число получается суммой длин градиентов которые указывают в этом направлении.

- 1) Инвариантность по повороту
- 2) Инвариантность по масштабу
- 3) Инвариантность по описанию:
 - К изменению яркости
 - К малому сдвигу



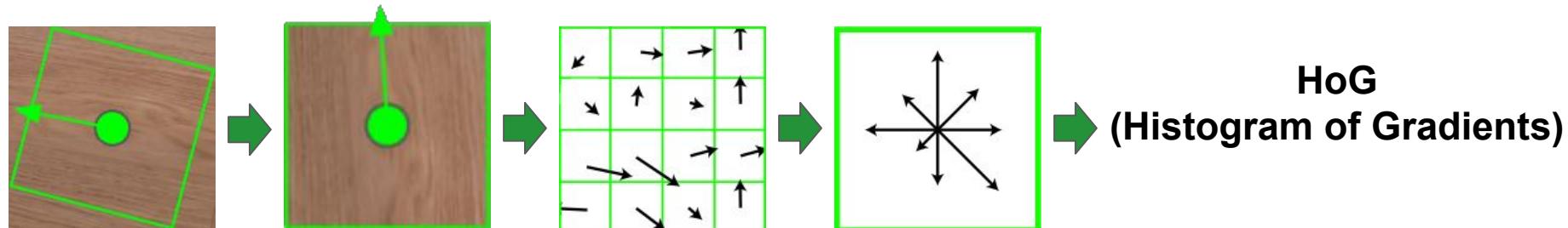
Key Points: Descriptor

Какие есть проблемы с **инвариантностью Descriptor-a?**

HoG = N чисел: одно число - одно направление.

Число получается суммой длин градиентов которые указывают в этом направлении.

- 1) Инвариантность по повороту
- 2) Инвариантность по масштабу
- 3) Инвариантность по описанию:
 - К изменению яркости
 - К малому сдвигу



SIFT Descriptor = HoG, т.е. 128-и корзинная гистограмма. В каждой корзине записано насколько популярно это направление.

P.S. на самом деле локальный патч бьется на мини-патчи, и строится **несколько HoG** - для каждого мини-патча. [Оригинальная статья](#).

Key Points: Descriptor

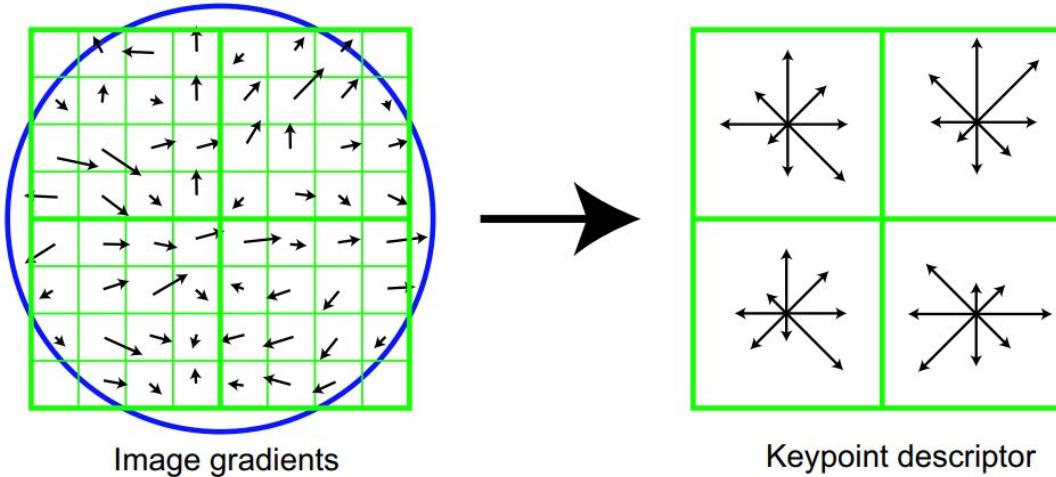


Figure 7: A keypoint descriptor is created by first computing the gradient magnitude and orientation at each image sample point in a region around the keypoint location, as shown on the left. These are weighted by a Gaussian window, indicated by the overlaid circle. These samples are then accumulated into orientation histograms summarizing the contents over 4x4 subregions, as shown on the right, with the length of each arrow corresponding to the sum of the gradient magnitudes near that direction within the region. This figure shows a 2x2 descriptor array computed from an 8x8 set of samples, whereas the experiments in this paper use 4x4 descriptors computed from a 16x16 sample array.

Р.С. на самом деле локальный патч бьется на мини-патчи, и строится несколько HoG - для каждого мини-патча. [Оригинальная статья](#).

Key Points: Descriptor

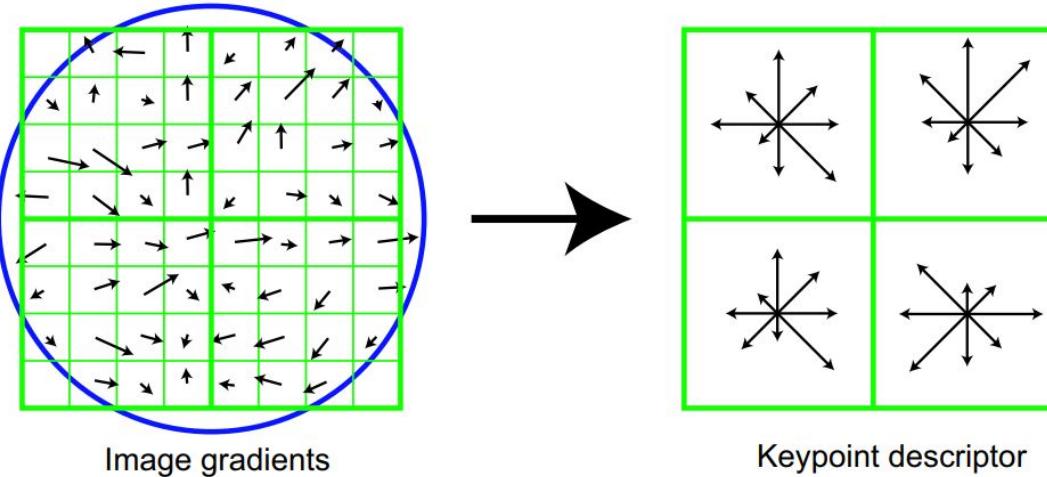


Figure 7: A keypoint descriptor is created by first computing the gradient magnitude and orientation at each image sample point in a region around the keypoint location, as shown on the left. These are weighted by a Gaussian window, indicated by the overlaid circle. These samples are then accumulated into orientation histograms summarizing the contents over 4x4 subregions, as shown on the right, with the length of each arrow corresponding to the sum of the gradient magnitudes near that direction within the region. This figure shows a 2x2 descriptor array computed from an 8x8 set of samples, whereas the experiments in this paper use 4x4 descriptors computed from a 16x16 sample array.

Р.С. на самом деле локальный патч бьется на мини-патчи, и строится **несколько HoG** - для каждого мини-патча. [Оригинальная статья](#).

Но раз SIFT Descriptor состоит из **128xFloat** и в каждой гистограмме **8xFloat**, то сколько мини-матчей проводит независимое голосование?

Key Points: Descriptor

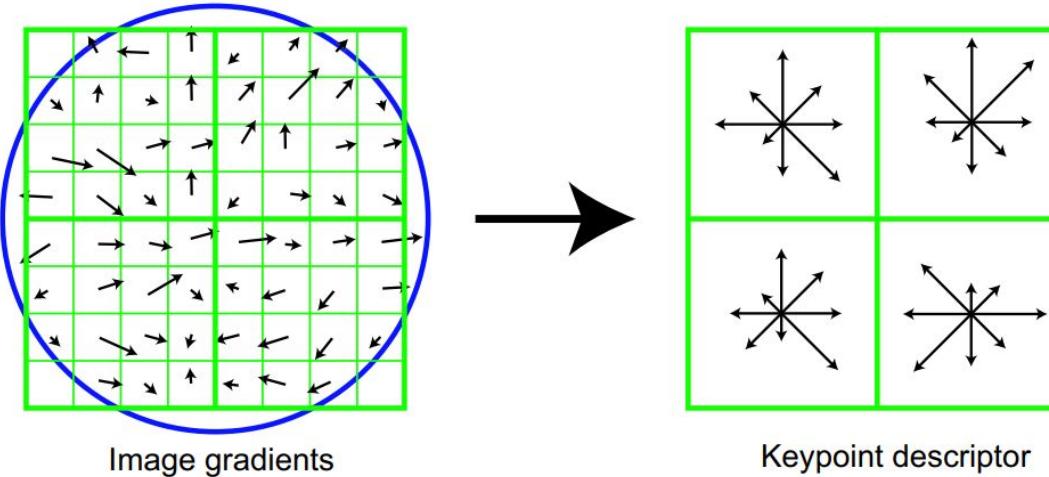


Figure 7: A keypoint descriptor is created by first computing the gradient magnitude and orientation at each image sample point in a region around the keypoint location, as shown on the left. These are weighted by a Gaussian window, indicated by the overlaid circle. These samples are then accumulated into orientation histograms summarizing the contents over 4x4 subregions, as shown on the right, with the length of each arrow corresponding to the sum of the gradient magnitudes near that direction within the region. This figure shows a 2x2 descriptor array computed from an 8x8 set of samples, whereas the experiments in this paper use 4x4 descriptors computed from a 16x16 sample array.

Р.С. на самом деле локальный патч бьется на мини-патчи, и строится **несколько HoG** - для каждого мини-патча. [Оригинальная статья](#).

Но раз SIFT Descriptor состоит из **128xFloat** и в каждой гистограмме **8xFloat**, то сколько мини-матчей проводит независимое голосование?

4x4! хоть слева и нарисовано 2x2.

Key Points: Descriptor

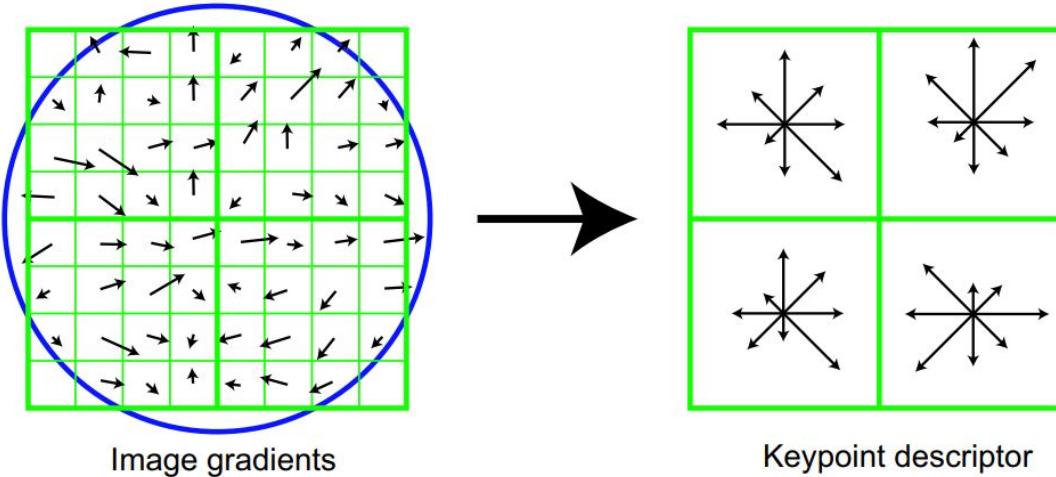


Figure 7: A keypoint descriptor is created by first computing the gradient magnitude and orientation at each image sample point in a region around the keypoint location, as shown on the left. These are weighted by a Gaussian window, indicated by the overlaid circle. These samples are then accumulated into orientation histograms summarizing the contents over 4×4 subregions, as shown on the right, with the length of each arrow corresponding to the sum of the gradient magnitudes near that direction within the region. This figure shows a 2×2 descriptor array computed from an 8×8 set of samples, whereas the experiments in this paper use 4×4 descriptors computed from a 16×16 sample array.

P.S. на самом деле локальный патч бьется на мини-патчи, и строится **несколько HoG** - для каждого мини-патча. [Оригинальная статья](#).

Но раз SIFT Descriptor состоит из **128xFloat** и в каждой гистограмме **8xFloat**, то сколько мини-матчей проводит независимое голосование?

4x4! хоть слева и нарисовано 2x2.

Затем эти **16** гистограмм по **8-корзин** конкатенируются в **$16 \times 8 = 128$** -мерный вектор.

Key Points: Descriptor

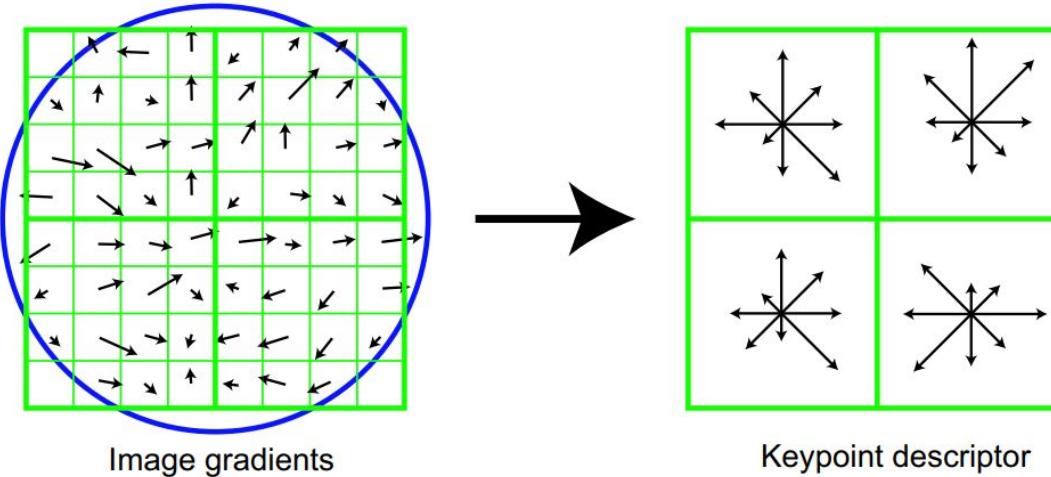


Figure 7: A keypoint descriptor is created by first computing the gradient magnitude and orientation at each image sample point in a region around the keypoint location, as shown on the left. These are weighted by a Gaussian window, indicated by the overlaid circle. These samples are then accumulated into orientation histograms summarizing the contents over 4x4 subregions, as shown on the right, with the length of each arrow corresponding to the sum of the gradient magnitudes near that direction within the region. This figure shows a 2x2 descriptor array computed from an 8x8 set of samples, whereas the experiments in this paper use 4x4 descriptors computed from a 16x16 sample array.

Р.С. на самом деле локальный патч бьется на мини-патчи, и строится **несколько HoG** - для каждого мини-патча. [Оригинальная статья](#).

Но раз SIFT Descriptor состоит из **128xFloat** и в каждой гистограмме **8xFloat**, то сколько мини-матчей проводит независимое голосование?

4x4! хоть слева и нарисовано 2x2.

Затем эти **16** гистограмм по **8-корзин** конкатенируются в **16x8=128**-мерный вектор.

Если яркость изображения везде увеличится на +100 - изменится ли дескриптор?

Key Points: Descriptor

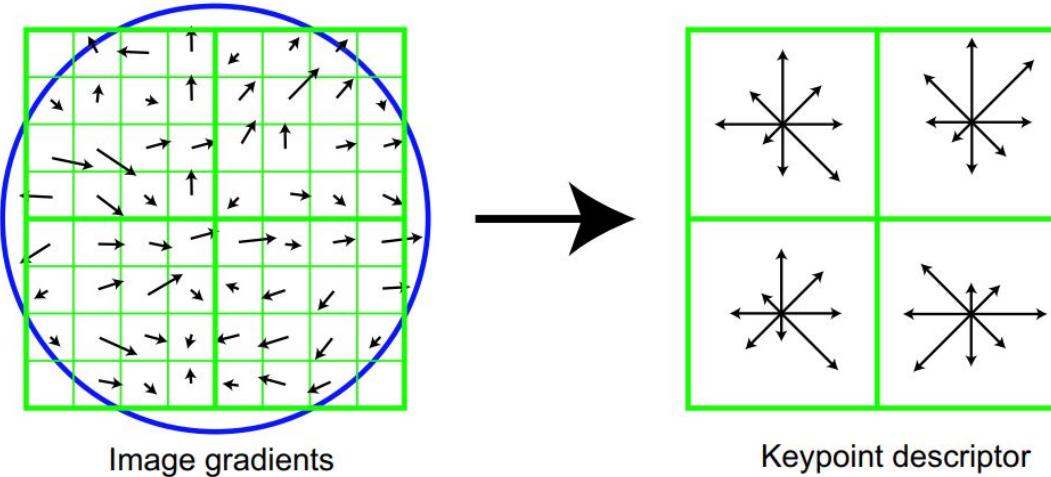


Figure 7: A keypoint descriptor is created by first computing the gradient magnitude and orientation at each image sample point in a region around the keypoint location, as shown on the left. These are weighted by a Gaussian window, indicated by the overlaid circle. These samples are then accumulated into orientation histograms summarizing the contents over 4x4 subregions, as shown on the right, with the length of each arrow corresponding to the sum of the gradient magnitudes near that direction within the region. This figure shows a 2x2 descriptor array computed from an 8x8 set of samples, whereas the experiments in this paper use 4x4 descriptors computed from a 16x16 sample array.

Р.С. на самом деле локальный патч бьется на мини-патчи, и строится **несколько HoG** - для каждого мини-патча. [Оригинальная статья](#).

Но раз SIFT Descriptor состоит из **128xFloat** и в каждой гистограмме **8xFloat**, то сколько мини-матчей проводит независимое голосование?

4x4! хоть слева и нарисовано 2x2.

Затем эти **16** гистограмм по **8-корзин** конкатенируются в **16x8=128**-мерный вектор.

Если яркость изображения везде умножится x2 - изменится ли дескриптор?

Key Points: Descriptor

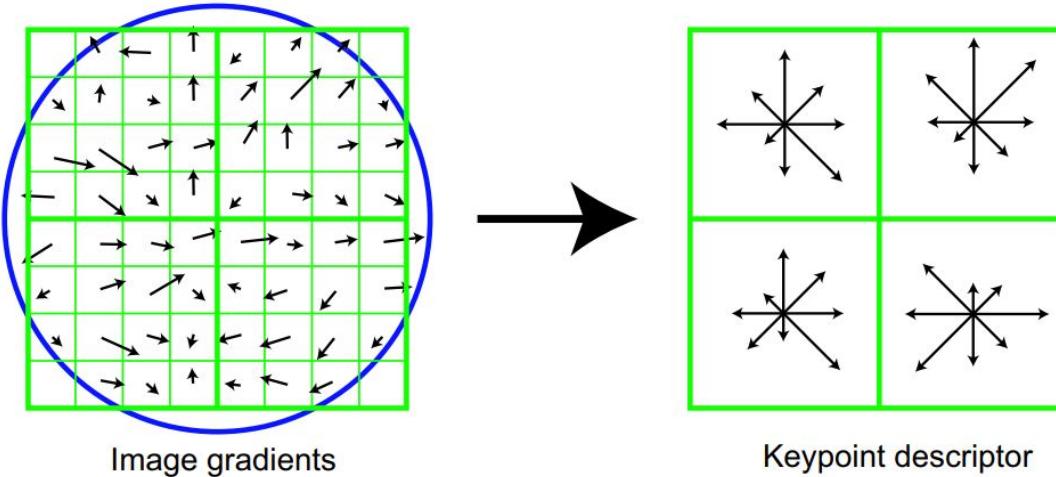


Figure 7: A keypoint descriptor is created by first computing the gradient magnitude and orientation at each image sample point in a region around the keypoint location, as shown on the left. These are weighted by a Gaussian window, indicated by the overlaid circle. These samples are then accumulated into orientation histograms summarizing the contents over 4×4 subregions, as shown on the right, with the length of each arrow corresponding to the sum of the gradient magnitudes near that direction within the region. This figure shows a 2×2 descriptor array computed from an 8×8 set of samples, whereas the experiments in this paper use 4×4 descriptors computed from a 16×16 sample array.

P.S. на самом деле локальный патч бьется на мини-патчи, и строится **несколько HoG** - для каждого мини-патча. [Оригинальная статья](#).

Но раз SIFT Descriptor состоит из **128xFloat** и в каждой гистограмме **8xFloat**, то сколько мини-матчей проводит независимое голосование?

4x4! хоть слева и нарисовано 2x2.

Затем эти **16** гистограмм по **8-корзин** конкатенируются в **$16 \times 8 = 128$** -мерный вектор.

Если яркость изображения везде умножится $x2$ - изменится ли дескриптор? Как исправить?

Key Points: Descriptor

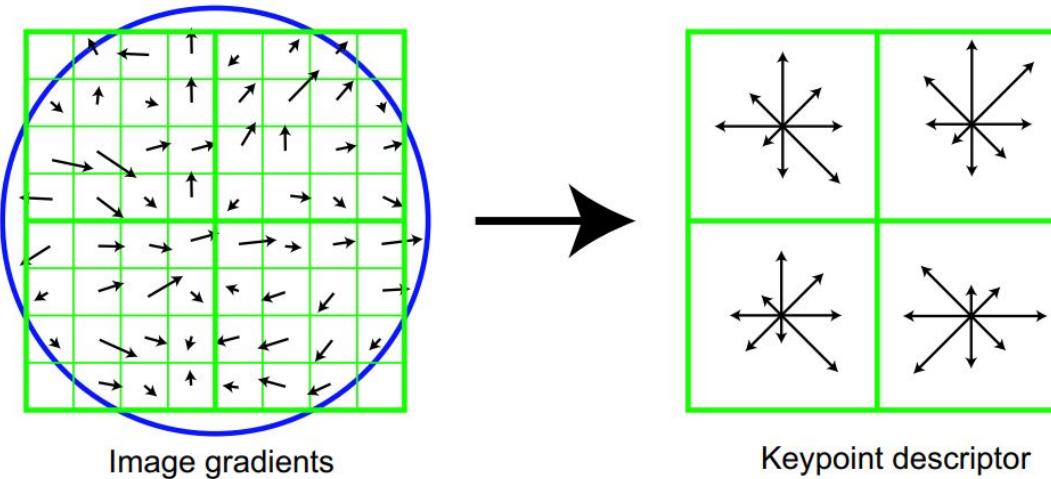


Figure 7: A keypoint descriptor is created by first computing the gradient magnitude and orientation at each image sample point in a region around the keypoint location, as shown on the left. These are weighted by a Gaussian window, indicated by the overlaid circle. These samples are then accumulated into orientation histograms summarizing the contents over 4x4 subregions, as shown on the right, with the length of each arrow corresponding to the sum of the gradient magnitudes near that direction within the region. This figure shows a 2x2 descriptor array computed from an 8x8 set of samples, whereas the experiments in this paper use 4x4 descriptors computed from a 16x16 sample array.

Р.С. на самом деле локальный патч бьется на мини-патчи, и строится **несколько HoG** - для каждого мини-патча. [Оригинальная статья](#).

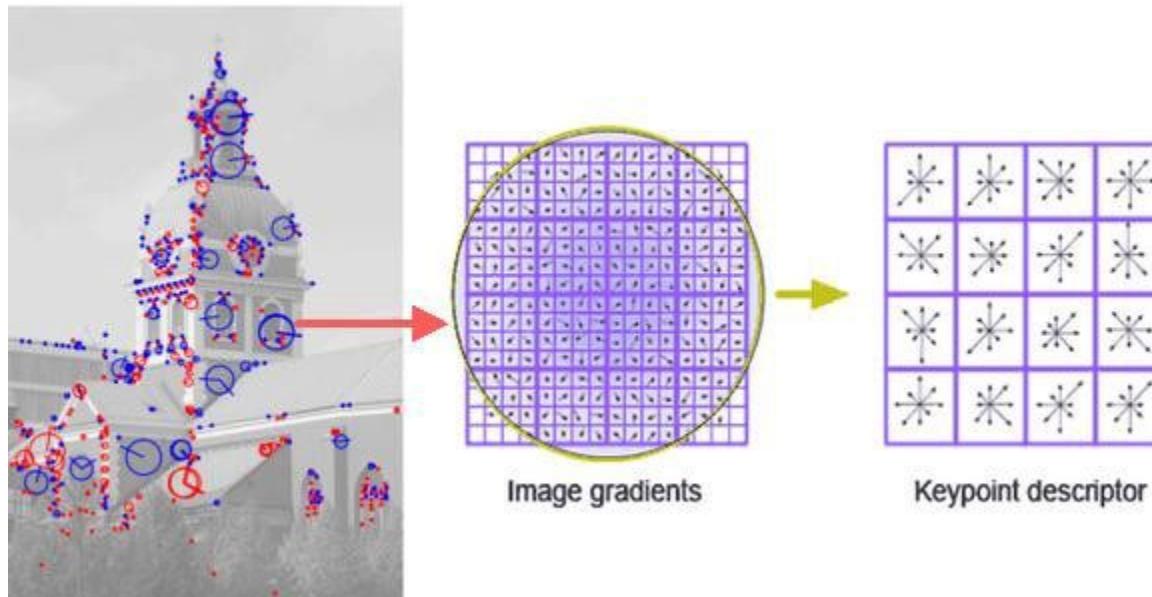
Но раз SIFT Descriptor состоит из **128xFloat** и в каждой гистограмме **8xFloat**, то сколько мини-матчей проводит независимое голосование?

4x4! хоть слева и нарисовано 2x2.

Затем эти **16** гистограмм по **8-корзин** конкатенируются в **16x8=128**-мерный вектор.

Наконец, **нормализация** до единичного вектора.

Key Points: Descriptor



Источник: <https://medium.com/machine-learning-world/feature-extraction-and-similar-image-search-with-opencv-for-newbies-3c59796bf774>

SIFT Key Points: Detector & Descriptor

- 1) Выбираем ключевыми точками “ пятна ”

SIFT Key Points: Detector & Descriptor

- 1) Выбираем ключевыми точками “ пятна ”
- 2) У каждой точки инвариантно определяем:
 - 2.1) Что?
 - 2.2) Что?

SIFT Key Points: Detector & Descriptor

- 1) Выбираем ключевыми точками “ пятна ”
- 2) У каждой точки инвариантно определяем:
 - 2.1) Поворот
 - 2.2) Масштаб

SIFT Key Points: Detector & Descriptor

- 1) Выбираем ключевыми точками “ пятна ”
- 2) У каждой точки инвариантно определяем:
 - 2.1) Поворот
 - 2.2) Масштаб

На базе какой информации мы можем это сделать ?

SIFT Key Points: Detector & Descriptor

- 1) Выбираем ключевыми точками “ пятна ”
- 2) У каждой точки **инвариантно** определяем (**на базе окрестности точки**):
 - 2.1) Поворот
 - 2.2) Масштаб

Что это? как она выглядит?



SIFT Key Points: Detector & Descriptor

- 1) Выбираем ключевыми точками “ пятна ”
- 2) У каждой точки **инвариантно** определяем (**на базе окрестности точки**):
 - 2.1) Поворот
 - 2.2) Масштаб

например квадратный патч 15x15 px



SIFT Key Points: Detector & Descriptor

- 1) Выбираем ключевыми точками “ пятна ”
- 2) У каждой точки **инвариантно** определяем (**на базе окрестности точки**):
 - 2.1) Поворот
 - 2.2) Масштаб
- 3) Для каждой точки строим дескриптор из **HoG** (гистограммы градиентов) по квадратной окрестности точки (**просто квадратный патч 15x15 px?**).

SIFT Key Points: Detector & Descriptor

- 1) Выбираем ключевыми точками “ пятна ”
- 2) У каждой точки **инвариантно** определяем (**на базе окрестности точки**):
 - 2.1) Поворот
 - 2.2) Масштаб
- 3) Для каждой точки строим дескриптор из **HoG** (гистограммы градиентов) по квадратной окрестности точки (поворнутой и отмасштабированной).

SIFT Key Points: Detector & Descriptor

- 1) Выбираем ключевыми точками “ пятна ”
- 2) У каждой точки **инвариантно** определяем (**на базе окрестности точки**):
 - 2.1) Поворот
 - 2.2) Масштаб
- 3) [+] Для каждой точки строим дескриптор из HoG (гистограммы градиентов) по квадратной окрестности точки (поворнутой и отмасштабированной).

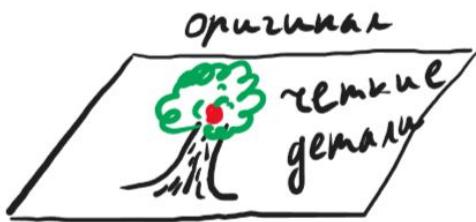
SIFT Key Points: Detector & Descriptor

- 1) Выбираем ключевыми точками “ пятна ”
- 2) У каждой точки **инвариантно** определяем (**на базе окрестности точки**):
 - 2.1) Поворот
 - 2.2) Масштаб
- 3) [+] Для каждой точки строим дескриптор из HoG (гистограммы градиентов) по квадратной окрестности точки (поворнутой и отмасштабированной).

SIFT Key Points: Detector & Descriptor

- 1) Выбираем ключевыми точками “ пятна ”
- 2) У каждой точки **инвариантно** определяем (**на базе окрестности точки**):
 - 2.1) **Поворот**
 - 2.2) **Масштаб**
- 3) [+] Для каждой точки строим дескриптор из HoG (гистограммы градиентов) по квадратной окрестности точки (поворнутой и отмасштабированной).

Разложим картинку на частоты



Разложим картинку на частоты

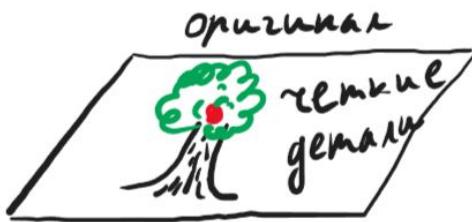


размытие (blur)

размытые
границы

A diagram showing the same tree drawing from the first frame, but with a blurry, out-of-focus effect applied to the edges. The words "размытые границы" are written next to it. A green arrow points downwards from the first frame to the second one.

Разложим картинку на частоты



размытие (blur)



размытие (blur)



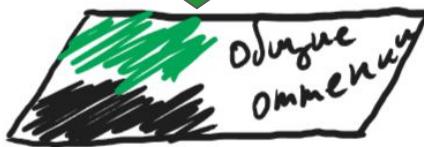
Разложим картинку на частоты



↓ размытие (blur)



↓ размытие (blur)



↓ размытие (blur)

что будет на **самом последнем** уровне???

Разложим картинку на частоты



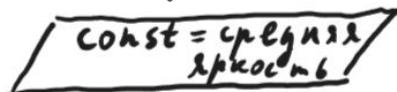
размытие (blur)



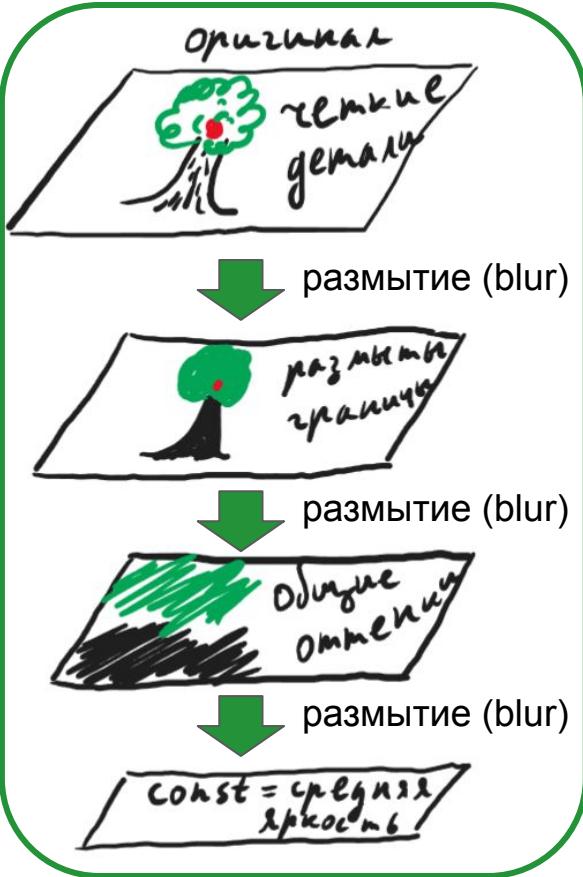
размытие (blur)



размытие (blur)

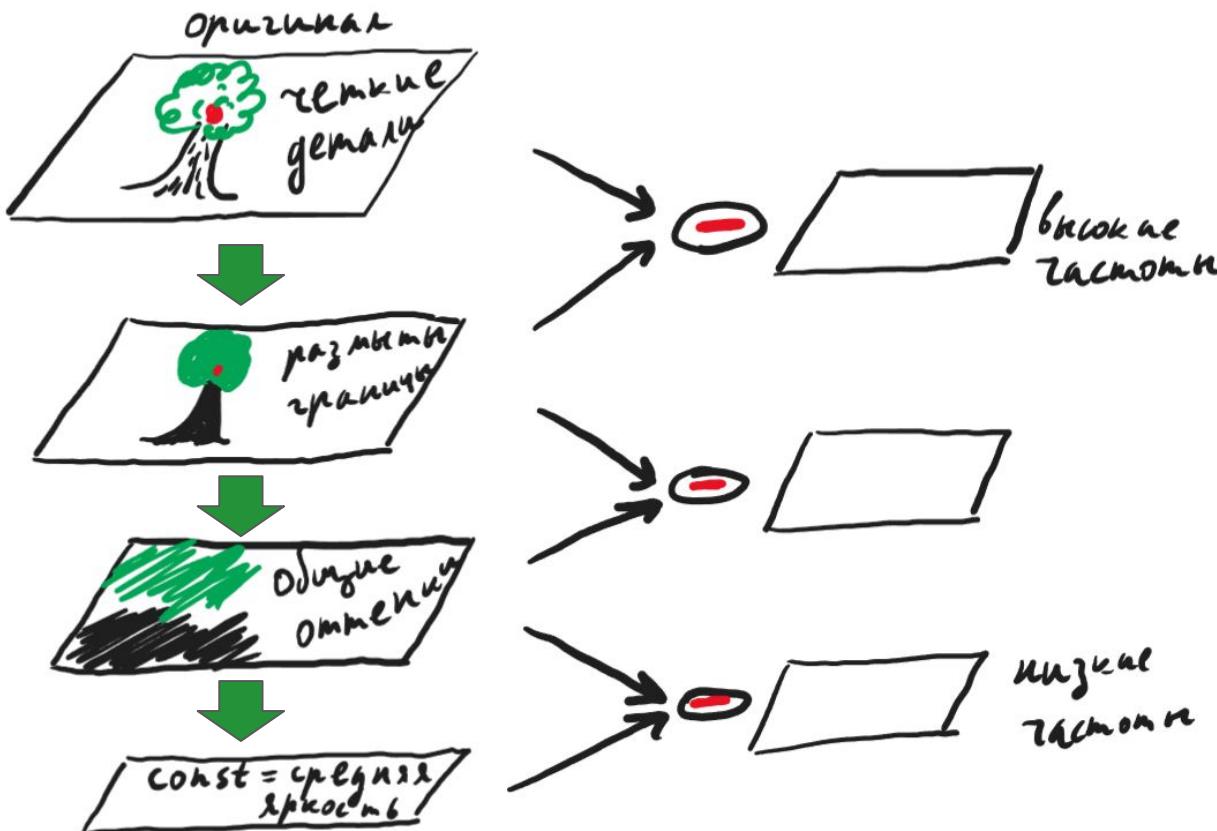


Разложим картинку на частоты



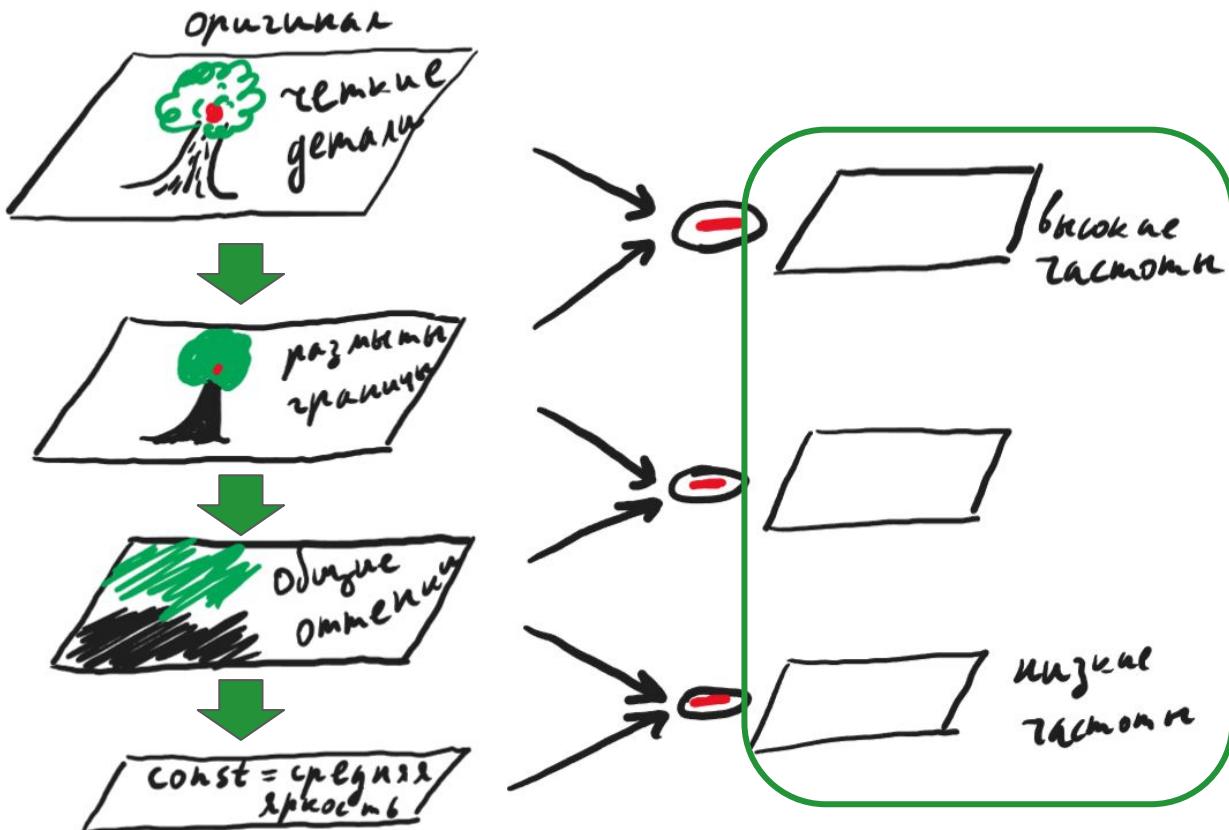
Gaussian Pyramid
(т.к. гауссово размытие)

Разложим картинку на частоты



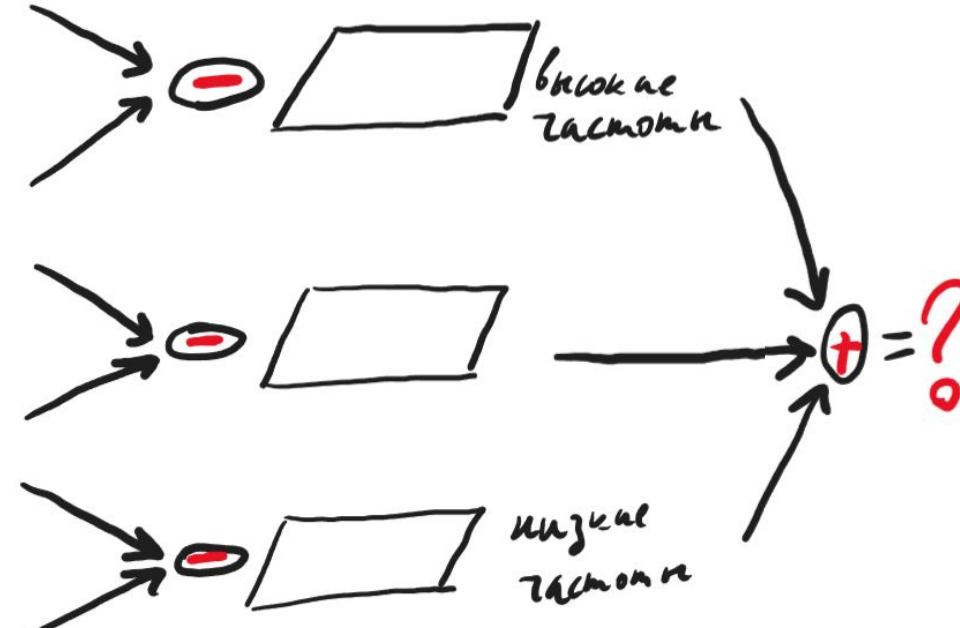
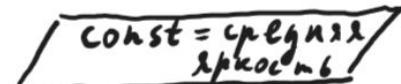
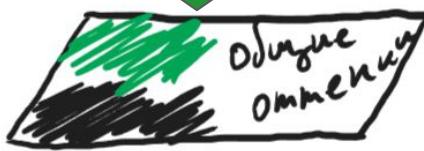
Gaussian Pyramid

Разложим картинку на частоты



Difference of Gaussian
(DoG)

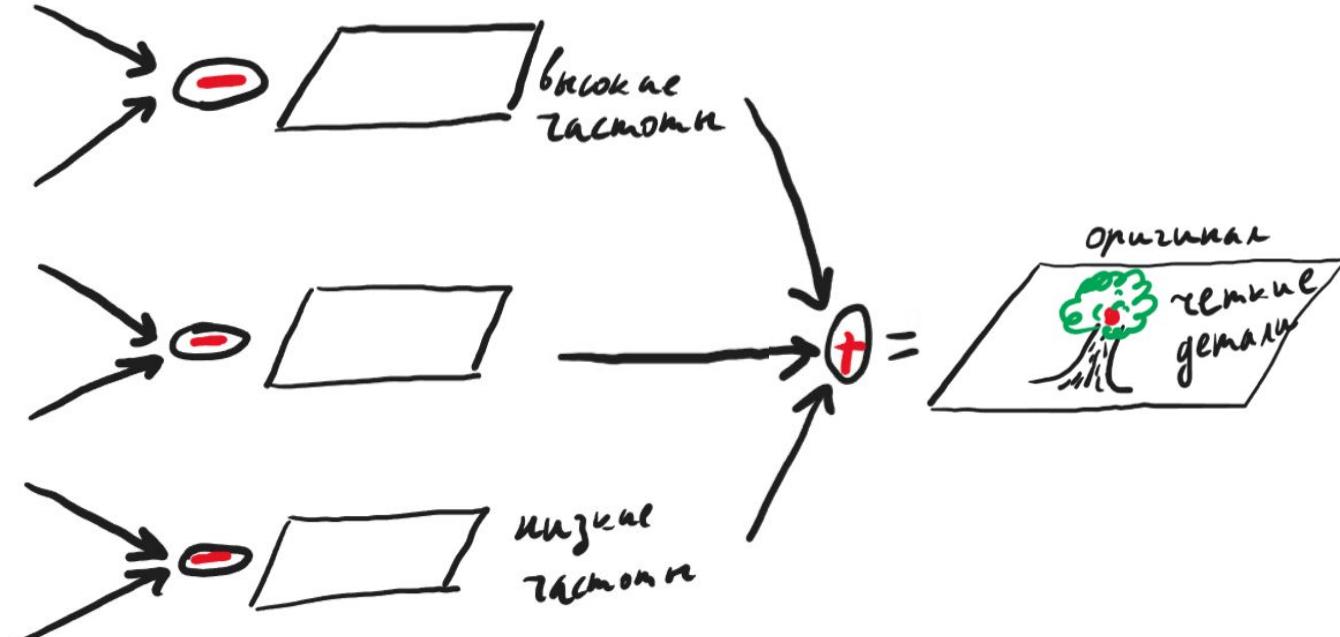
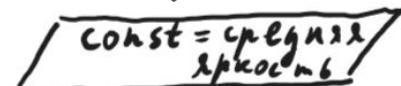
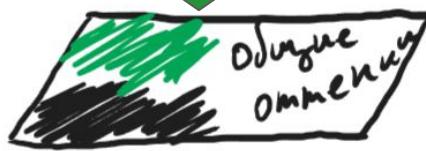
Разложим картинку на частоты



Gaussian Pyramid

Difference of Gaussian
(DoG)

Разложим картинку на частоты



Gaussian Pyramid

Difference of Gaussian
(DoG)

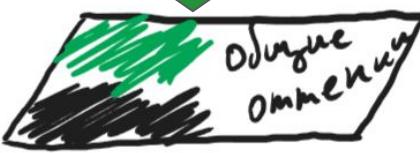
Разложим картинку на частоты



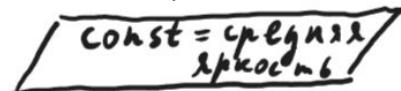
размытие (blur)



размытие (blur)



размытие (blur)



Gaussian Pyramid

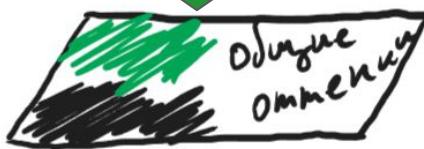
Разложим картинку на частоты



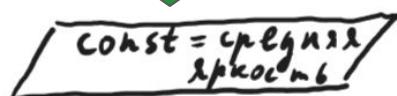
размытие (blur)



размытие (blur)

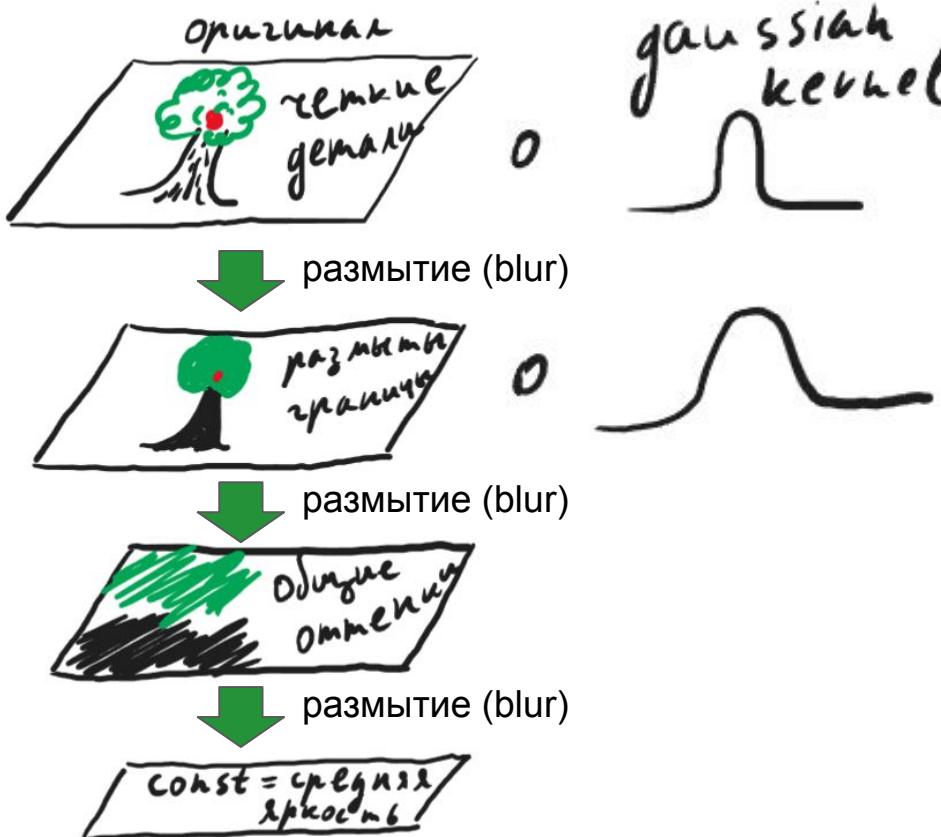


размытие (blur)



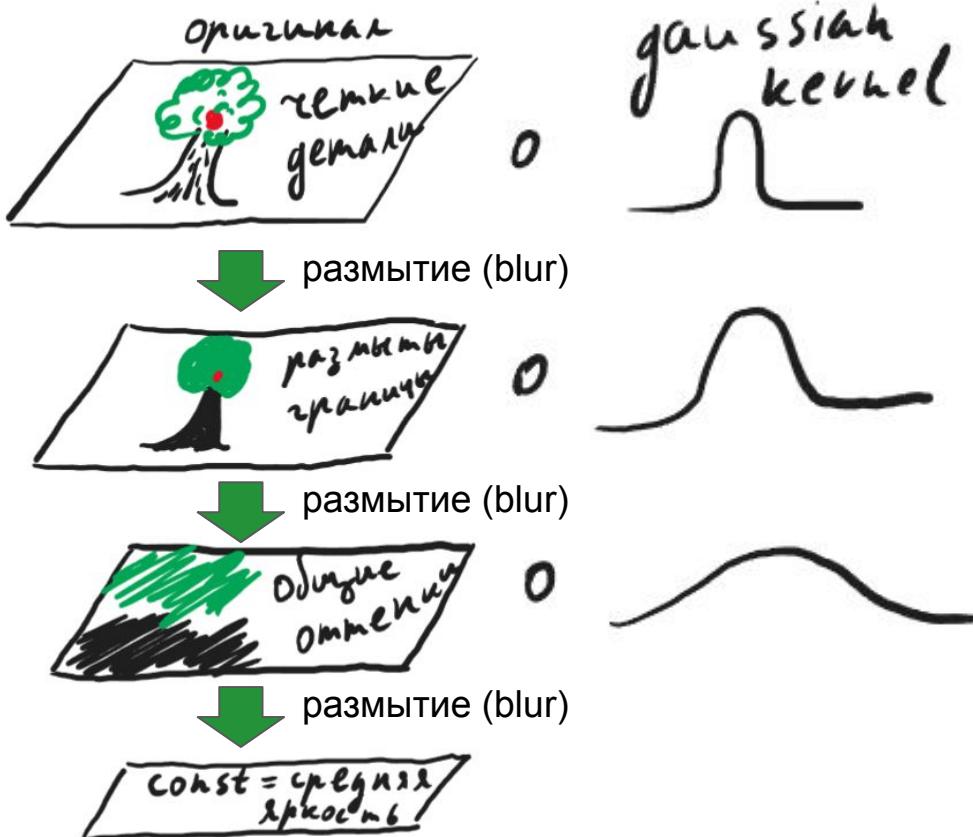
Gaussian Pyramid

Разложим картинку на частоты



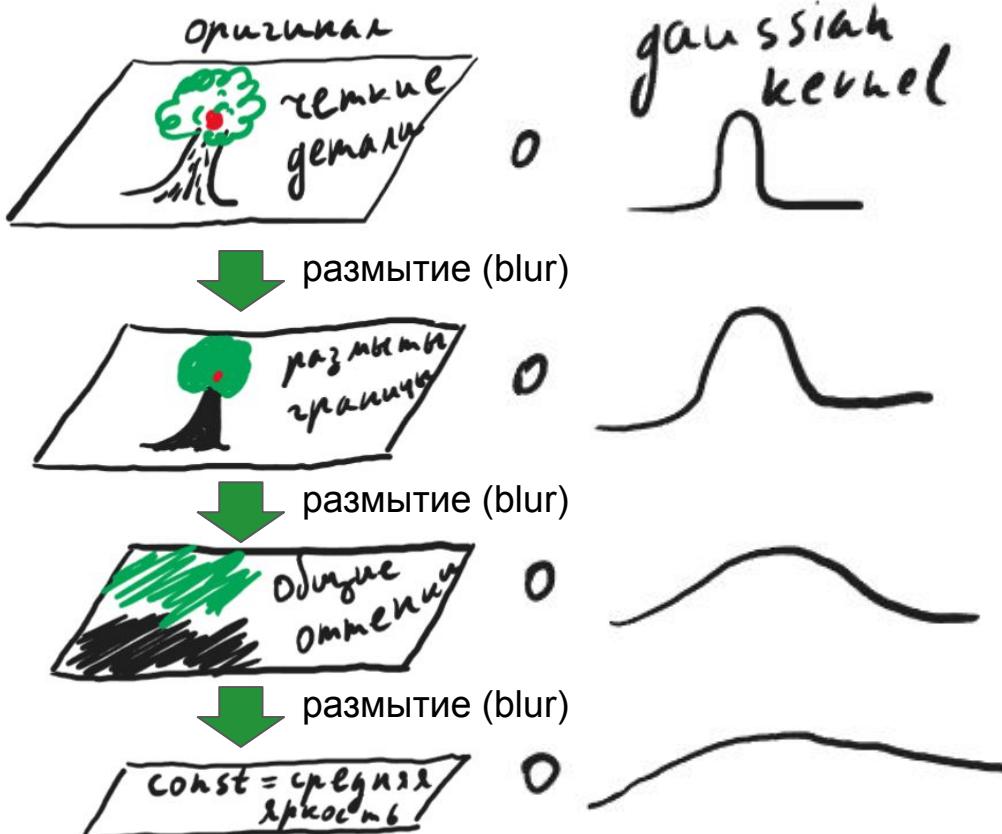
- фильтрует (убирает) **высокие частоты** (т.е. частоты выше некоторого порога)
например **шум**

Разложим картинку на частоты



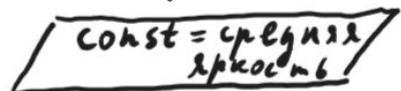
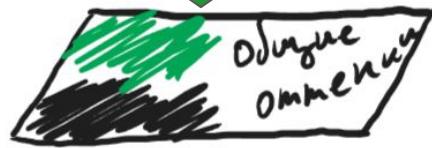
- фильтрует (убирает) **высокие частоты** (т.е. частоты выше некоторого порога)
например **шум**
- фильтрует (убирает) **средние частоты** (т.е. частоты выше некоторого порога)

Разложим картинку на частоты



- фильтрует (убирает) **высокие частоты** (т.е. частоты выше некоторого порога)
например **шум**
- фильтрует (убирает) **средние частоты** (т.е. частоты выше некоторого порога)
- фильтрует (убирает) **почти все частоты** кроме уже совсем низких

Разложим картинку на частоты



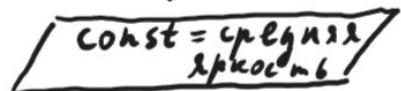
Gaussian Pyramid

убраны высокие частоты

убраны высокие частоты и средние частоты

Difference of Gaussian
(DoG)

Разложим картинку на частоты



Gaussian Pyramid

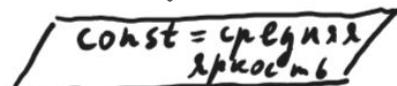
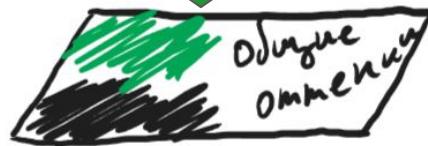
Difference of Gaussian
(DoG)

убраны высокие частоты

???

убраны высокие частоты и средние частоты

Разложим картинку на частоты



Gaussian Pyramid

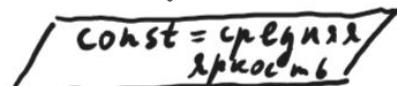
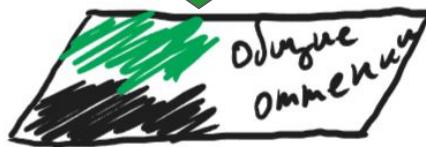
Difference of Gaussian
(DoG)

убраны высокие частоты

высоких частот не было ни сверху ни снизу

убраны высокие частоты и средние частоты

Разложим картинку на частоты



Gaussian Pyramid

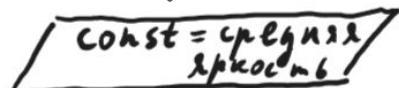
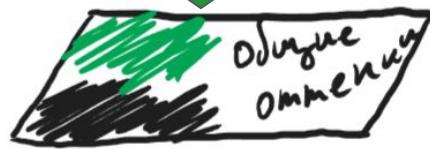
убраны высокие частоты

высоких частот не было ни сверху ни снизу
низкие частоты были и там и там - сократились

убраны высокие частоты и средние частоты

Difference of Gaussian
(DoG)

Разложим картинку на частоты



убраны высокие частоты

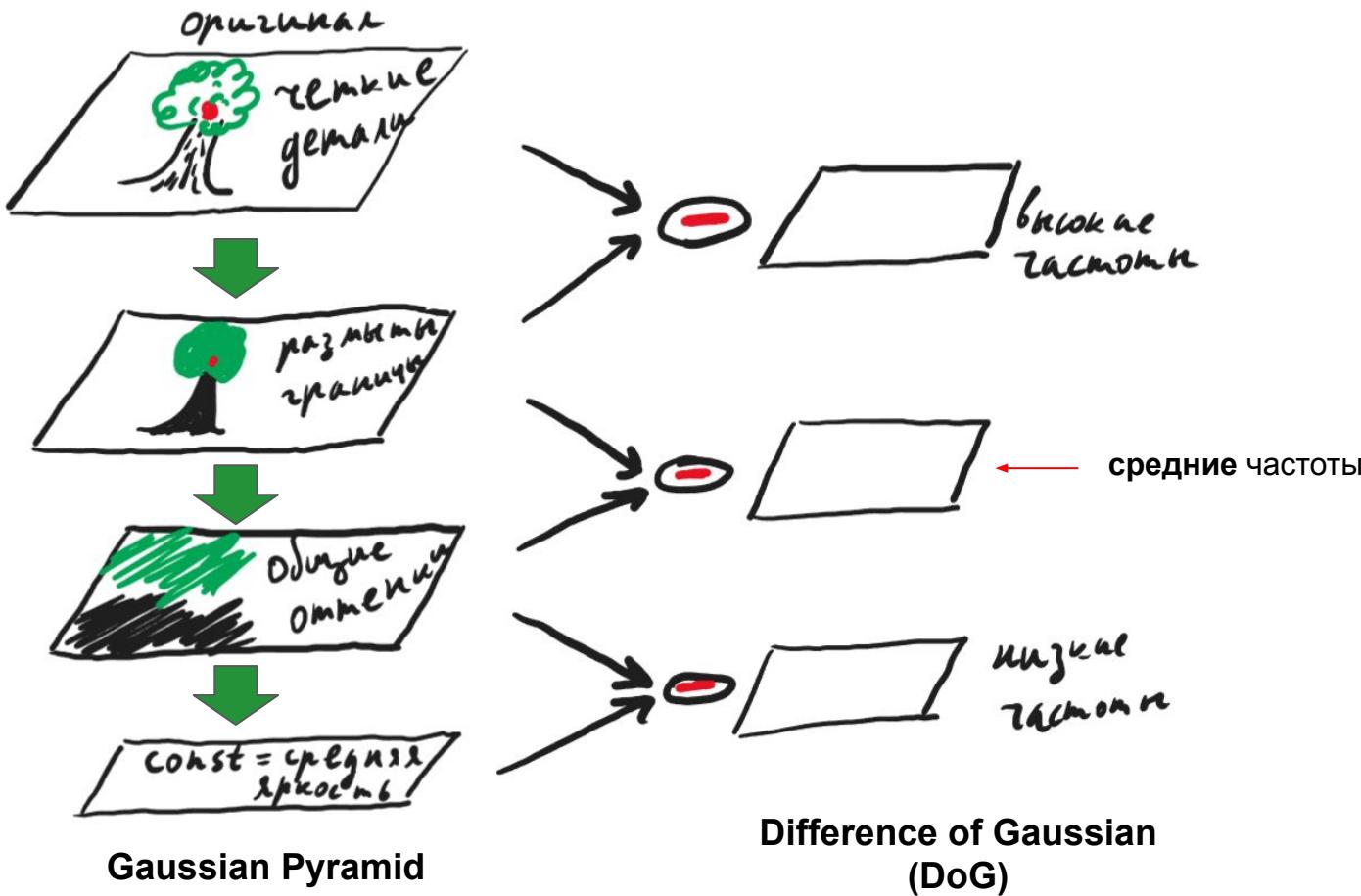
высоких частот не было ни сверху ни снизу
низкие частоты были и там и там - сократились
остались только **средние** частоты

убраны высокие частоты и средние частоты

Gaussian Pyramid

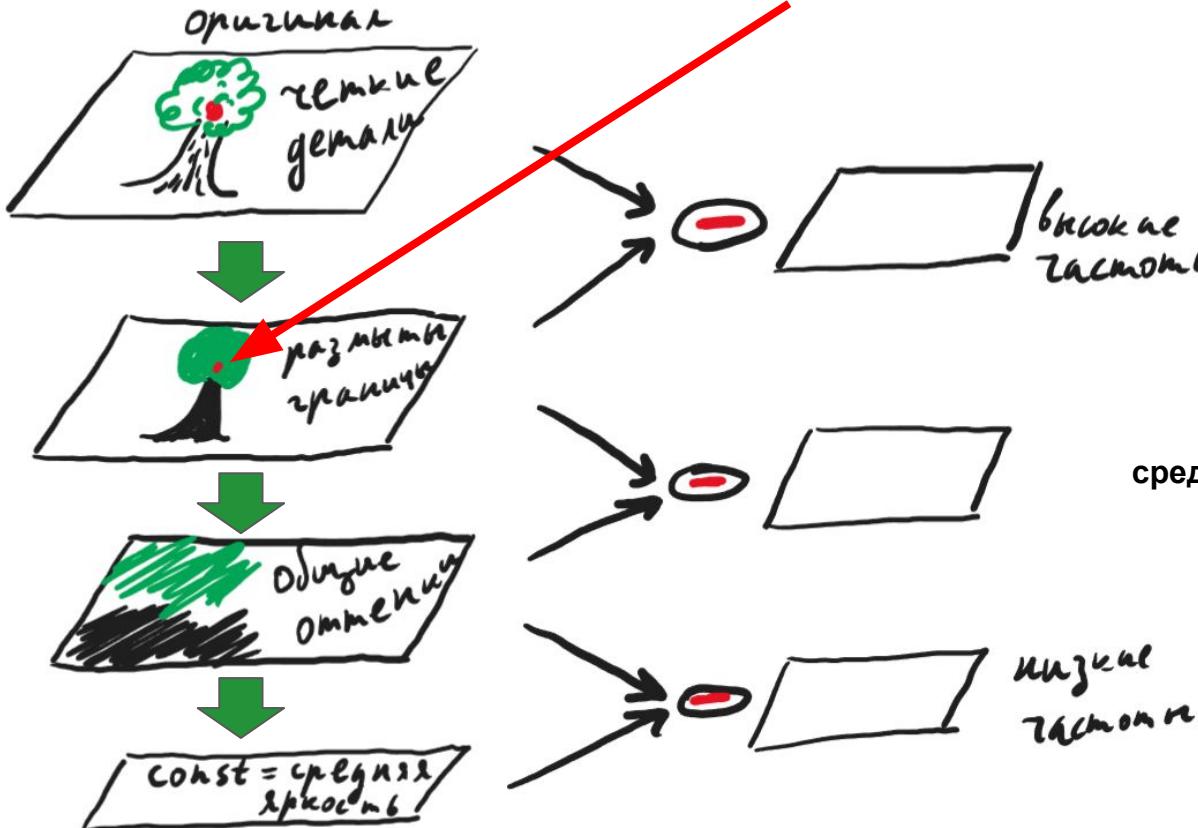
Difference of Gaussian
(DoG)

Разложим картинку на частоты



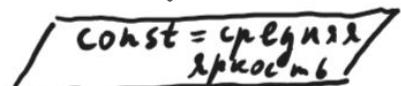
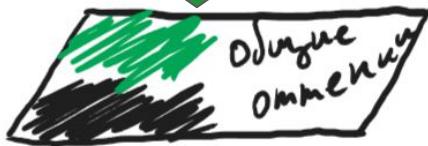
Разложим картинку на частоты

на каком слое частот мы можем увидеть след от яблока?



Difference of Gaussian
(DoG)

Разложим картинку на частоты



если яблоко убрать - какие частоты:

- почти не изменяются?
- изменяется сильно?
- слабо изменяется?

высокие частоты

средние частоты

низкие частоты

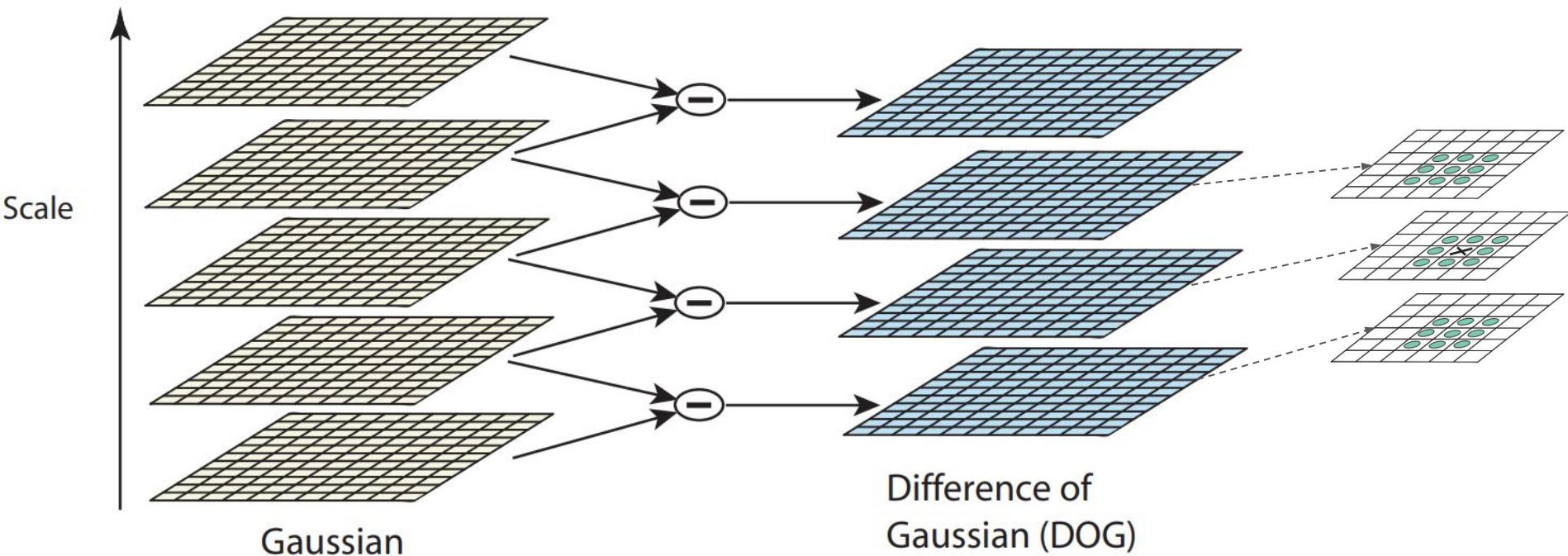
Gaussian Pyramid

Difference of Gaussian
(DoG)

SIFT Key Points: Detector

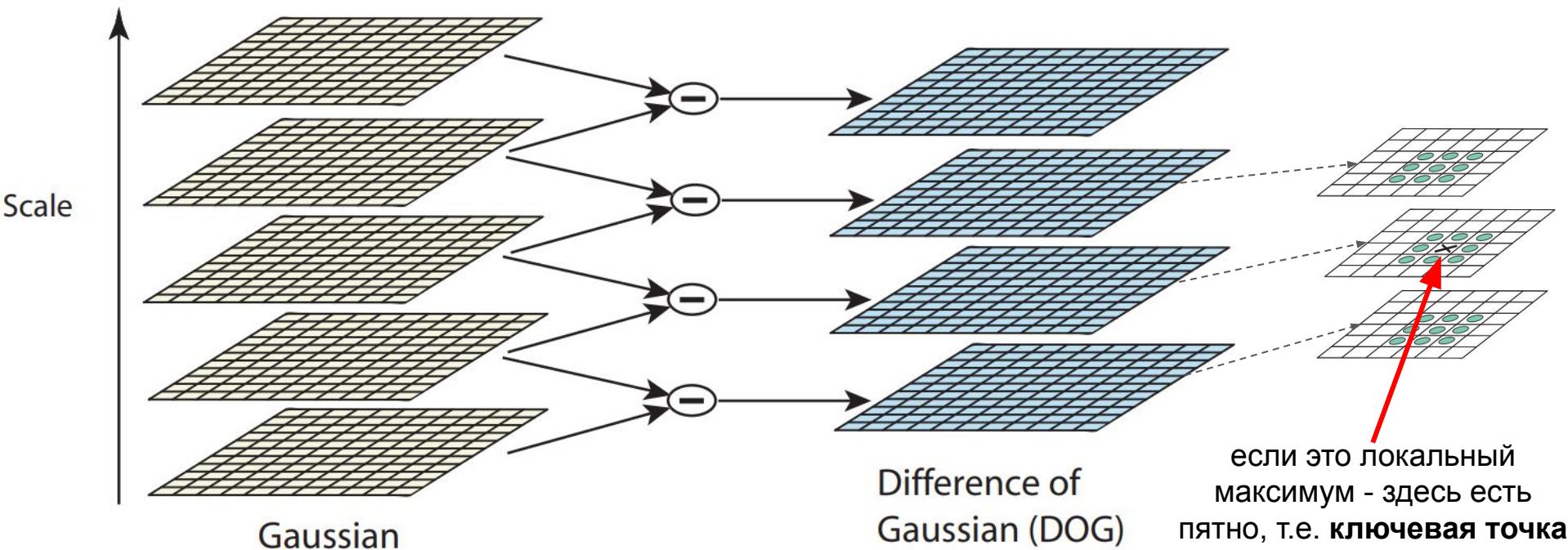
- 1) Выбираем ключевыми точками “ пятна ”
- 2) У каждой точки **инвариантно** определяем (**на базе окрестности точки**):
 - 2.1) Поворот**
 - 2.2) Масштаб**

Разложим картинку на частоты



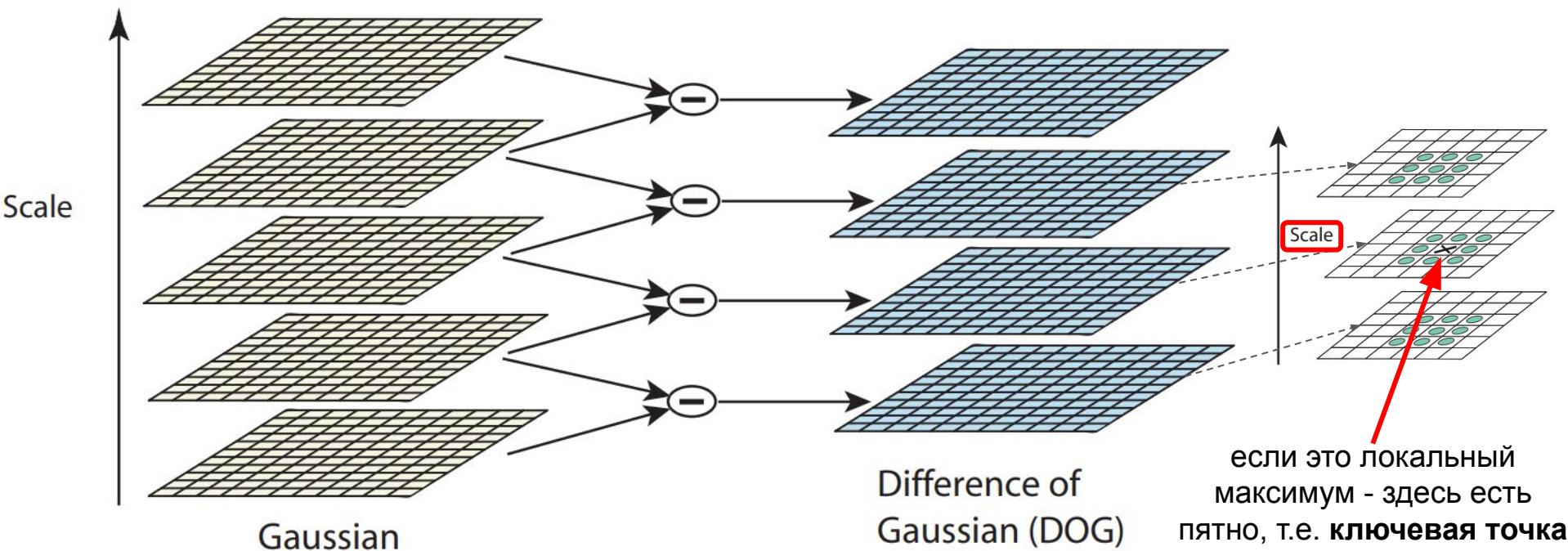
Вот такая иллюстрация есть в статье про SIFT Detector... Что она там делает?

Разложим картинку на частоты



Вот такая иллюстрация есть в статье про SIFT Detector... Что она там делает?

Разложим картинку на частоты



Вот такая иллюстрация есть в статье про SIFT Detector... Что она там делает?

SIFT Key Points: Detector

- 1) Выбираем ключевыми точками “ пятна ”
- 2) У каждой точки **инвариантно** определяем (**на базе окрестности точки**):
 - 2.1) Поворот
 - 2.2) **Масштаб**

SIFT Key Points: Detector

- 1) Выбираем ключевыми точками “ пятна ”
- 2) У каждой точки **инвариантно** определяем (**на базе окрестности точки**):
 - 2.1) Поворот
 - 2.2) **Масштаб**

Чем хорошо такое определение масштаба?

Почему нельзя его всегда брать например за 15 пикселей?

SIFT Key Points: Detector

- 1) Выбираем ключевыми точками “ пятна ”
- 2) У каждой точки **инвариантно** определяем (**на базе окрестности точки**):
 - 2.1) Поворот
 - 2.2) **Масштаб**

Инвариантность!

Благодаря этому масштаб зависит от окрестности точки в реальном мире, т.е. от ее “размеров пятна в реальном мире” и проецирования в фотографию.

SIFT Key Points: Detector

- 1) Выбираем ключевыми точками “ пятна ”
- 2) У каждой точки **инвариантно** определяем (**на базе окрестности точки**):
 - 2.1) Поворот
 - 2.2) **Масштаб**

Инвариантность!

Благодаря этому масштаб зависит от окрестности точки в реальном мире, т.е. от ее “размеров пятна в реальном мире” и проецирования в фотографию.

А значит в другой фотографии размер будет такой же по семантике (содержимому окрестности этого масштаба), хоть и другой в пикселях.

SIFT Key Points: Detector

- 1) Выбираем ключевыми точками “ пятна ”
- 2) У каждой точки **инвариантно** определяем (**на базе окрестности точки**):
 - 2.1) **Поворот**
 - 2.2) **[+]** **Масштаб**

А как инвариантно определить поворот?

SIFT Key Points: Detector

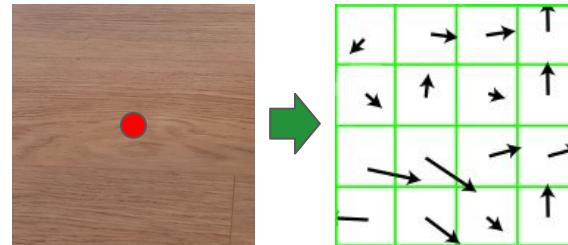
- 1) Выбираем ключевыми точками “ пятна ”
- 2) У каждой точки **инвариантно** определяем (**на базе окрестности точки**):
 - 2.1) **Поворот**
 - 2.2) **[+]** **Масштаб**

А как инвариантно определить поворот?

Чтобы в другой фотографии у этой точки был такое же направление в системе координат мира.

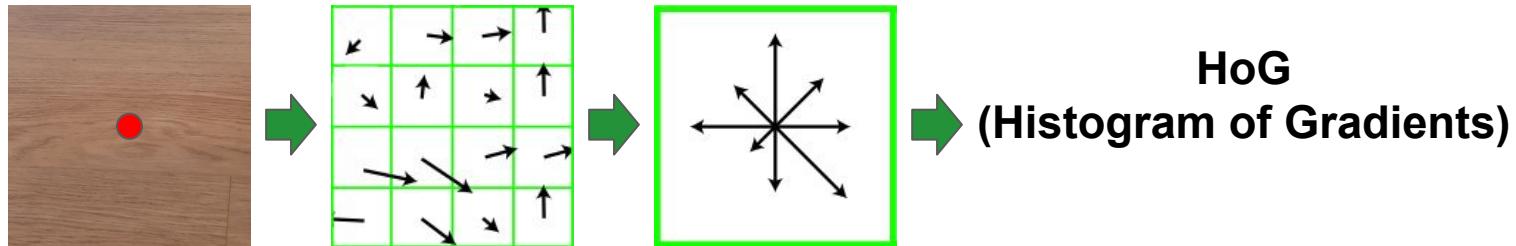
SIFT Key Points: Detector

- 1) Выбираем ключевыми точками “ пятна ”
- 2) У каждой точки **инвариантно** определяем (**на базе окрестности точки**):
 - 2.1) **Поворот**
 - 2.2) **[+]** **Масштаб**



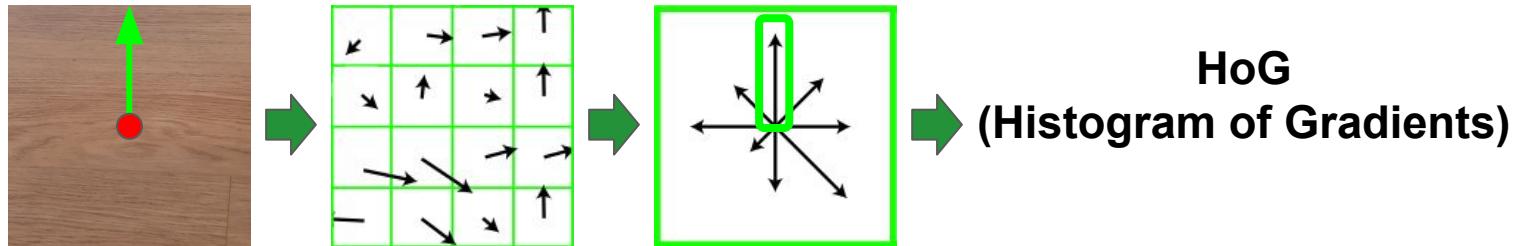
SIFT Key Points: Detector

- 1) Выбираем ключевыми точками “ пятна ”
- 2) У каждой точки **инвариантно** определяем (**на базе окрестности точки**):
 - 2.1) **Поворот**
 - 2.2) **[+]** **Масштаб**



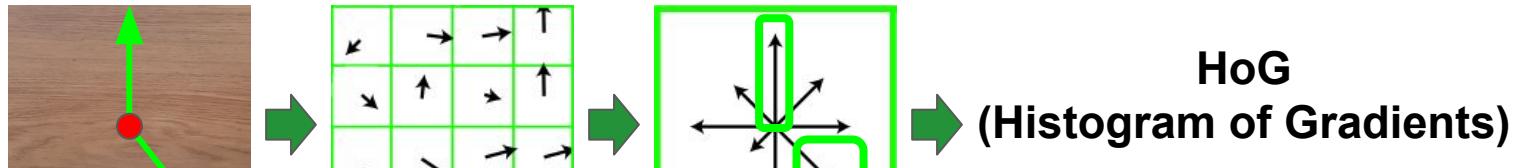
SIFT Key Points: Detector

- 1) Выбираем ключевыми точками “ пятна ”
- 2) У каждой точки **инвариантно** определяем (**на базе окрестности точки**):
 - 2.1) **Поворот**
 - 2.2) **[+]** **Масштаб**



SIFT Key Points: Detector

- 1) Выбираем ключевыми точками “ пятна ”
- 2) У каждой точки **инвариантно** определяем (**на базе окрестности точки**):
 - 2.1) **Поворот**
 - 2.2) **[+]** **Масштаб**



Что можно сделать если два (или
больше) очень близких максимума?

SIFT Key Points: Detector

- 1) Выбираем ключевыми точками “ пятна ”
- 2) У каждой точки **инвариантно** определяем (**на базе окрестности точки**):
 - 2.1) **[+]** Поворот
 - 2.2) **[+]** Масштаб



SIFT Key Points: Detector & Descriptor

- 1) Выбираем ключевыми точками “ пятна ”

SIFT Key Points: Detector & Descriptor

- 1) Выбираем ключевыми точками “ пятна ”
- 2) У каждой точки **инвариантно** определяем (**на базе окрестности точки**):
 - 2.1) Поворот
 - 2.2) Масштаб

SIFT Key Points: Detector & Descriptor

- 1) Выбираем ключевыми точками “ пятна ”
- 2) У каждой точки **инвариантно** определяем (**на базе окрестности точки**):
 - 2.1) Поворот
 - 2.2) Масштаб
- 3) Для каждой точки строим дескриптор из **HoG** (гистограммы градиентов) по квадратной окрестности точки (поворнутой и отмасштабированной)

Вопрос 1: как обнаружать экстремум еще точнее?

Вопрос 2: как обнаружать поворот точки точнее?

Вопрос 3: как ускорить алгоритм детектирования?

Вопрос 4: что в статье делается по-другому?

Вопрос 5: на что влияет размер патча?

Вопрос 6: есть ли проблемы из-за квадратности окрестности при построении дескриптора? Как исправить?

Вопрос 7: какой этап занимает больше всего времени?

Вопрос 8: какой Detector&Descriptor для 3D облака точек?

SIFT. Ссылки

Задание: <https://github.com/PhotogrammetryCourse>

SIFT: [Distinctive Image Features from Scale-Invariant Keypoints, G. Lowe, 2004](#)

Примеры реализации: [OpenSIFT](#) и его адаптации - [by lxc-xx](#) и в OpenCV ([sift.dispatch.cpp](#), [sift.simd.hpp](#)).

Сравнения разных детекторов и дескрипторов:

- 1) [Local Invariant Feature Detectors: A Survey, T. Tuytelaars, K. Mikolajczyk, 2008](#)
- 2) [A survey of recent advances in visual feature detection, Y. Li et al., 2015](#)
- 3) [Recent Advances in Features Extraction and Description Algorithms: A Comprehensive Survey, E. Salahat, M. Qasaimeh, 2017](#)

Очень советую <https://www.semanticscholar.org/> для того чтобы найдя одну хорошую статью - идти по цепочкам цитат как в статьи от которых эта статья отталкивалась, так и в статьи которые ссылаются на эту статью.

Вопросы?



Полярный Николай
polarnick239@gmail.com