

Введение в фотограмметрию

Построение карт глубины

Метод Patch Match

Фотограмметрия. Лекция 12

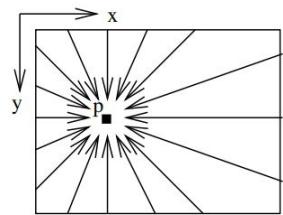


- Карты глубины + нормалей
- Учет всех картинок
- Patch Match (ретуширование картинки)

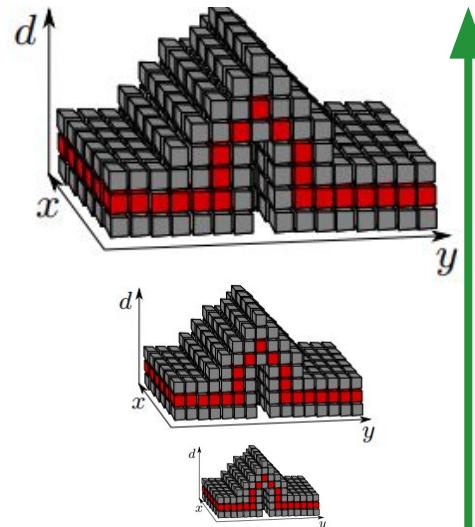
Полярный Николай
polarnick239@gmail.com

Карта диспаритета - Semi-Global Matching (**SGM**, **tSGM**)

(b) 16 Paths from all Directions

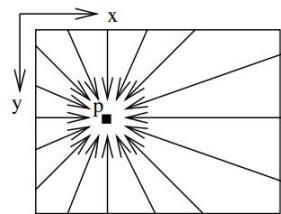


За счет **ректификации** стереопары и сведения к задаче **динамического программирования** (агрегация по 16 направлениям) - довольно быстро работает.

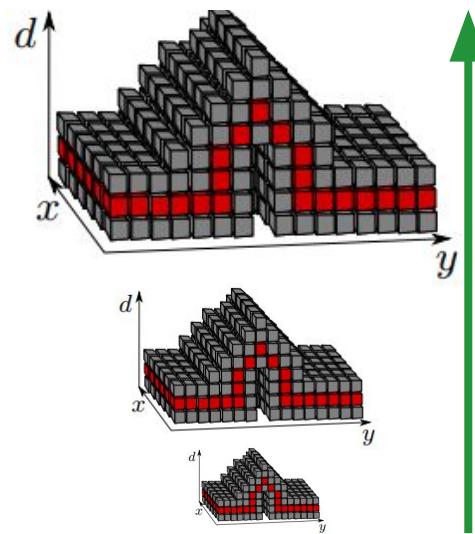


Карта диспаритета - Semi-Global Matching (**SGM**, **tSGM**)

(b) 16 Paths from all Directions r



За счет **ректификации** стереопары и сведения к задаче **динамического программирования** (агрегация по 16 направлениям) - довольно быстро работает.



За счет **coarse-to-fine** схемы (**tSGM**: пирамиды картинок + прогрессивный поиск карты глубины на новой ступени детализации в ограниченном диапазоне диспаритетов) - еще быстрее и, главное, с гарантиями по памяти.

Попробуем придумать альтернативный метод?

Итого:

- Есть точная калибровка камеры и ракурсов фотографирования (**intrinsics & extrinsics parameters**)
- Есть точное но разреженное (недетальное) облако 3D точек

Хотим:

- Детальную геометрию - например плотное облако точек:
один пиксель - одна точка

Какие есть проблемы в SGM?

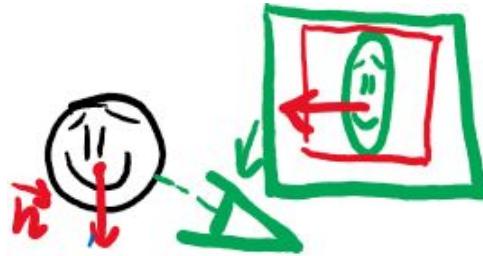
Что плохо в SGM?

Иначе говоря - какие есть точки роста в качестве результата?

Из-за чего в алгоритме что-то было плохо, а значит критически важно это учесть чтобы придумать новый метод в котором этот недостаток будет учтен?

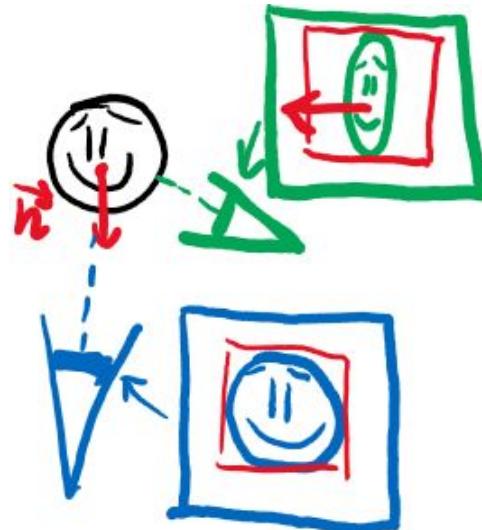
Какие есть проблемы в **SGM**?

- 1) В **SGM** мы предполагаем что все патчи - фронтопараллельные. А значит как только углы наблюдения далеки от 90 градусов - мы ошибаемся.



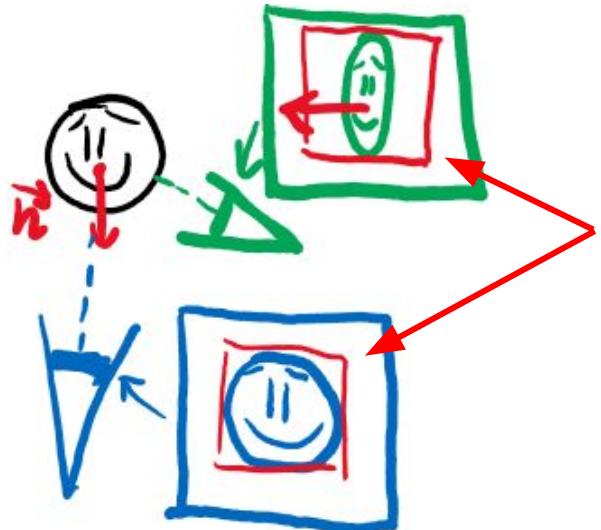
Какие есть проблемы в **SGM**?

- 1) В **SGM** мы предполагаем что все патчи - фронтопараллельные. А значит как только углы наблюдения далеки от 90 градусов - мы ошибаемся.



Какие есть проблемы в SGM?

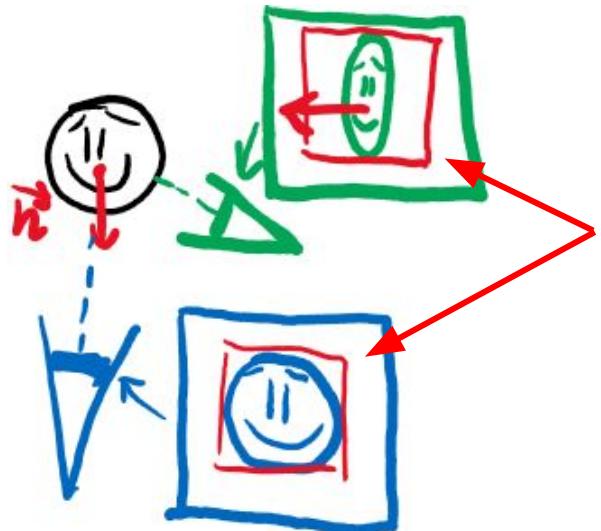
- 1) В SGM мы предполагаем что все патчи - фрonto-параллельные. А значит как только углы наблюдения далеки от 90 градусов - мы ошибаемся.



Нельзя просто брать и сравнивать ZNCC/Census в некотором фиксированном по размеру окне, т.к. поверхность может быть **под углом**.

Какие есть проблемы в SGM?

- 1) В SGM мы предполагаем что все патчи - фрonto-параллельные. А значит как только углы наблюдения далеки от 90 градусов - мы ошибаемся.

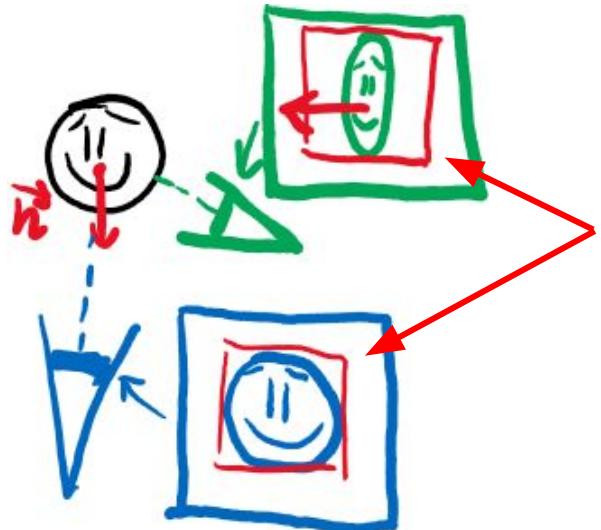


Нельзя просто брать и сравнивать ZNCC/Census в некотором фиксированном по размеру окне, т.к. поверхность может быть **под углом**.

Как это исправить?

Какие есть проблемы в SGM?

- 1) В SGM мы предполагаем что все патчи - фрonto-параллельные. А значит как только углы наблюдения далеки от 90 градусов - мы ошибаемся.

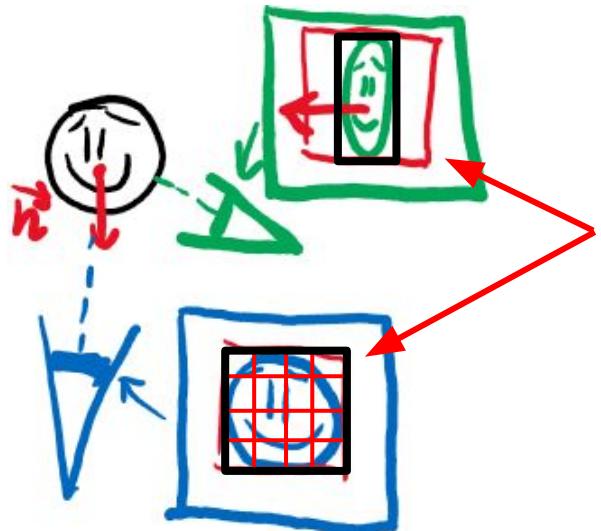


Нельзя просто брать и сравнивать ZNCC/Census в некотором фиксированном по размеру окне, т.к. поверхность может быть **под углом**.

Как это исправить если нам известно не только значение диспаритета, но и нормаль поверхности n ?

Какие есть проблемы в SGM?

- 1) В SGM мы предполагаем что все патчи - фрonto-параллельные. А значит как только углы наблюдения далеки от 90 градусов - мы ошибаемся.

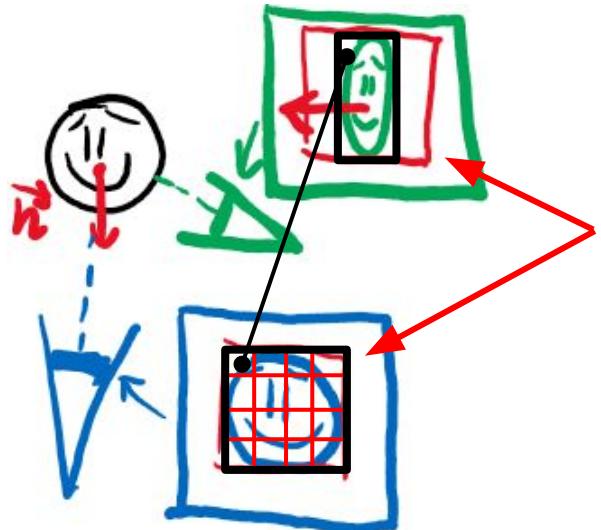


Нельзя просто брать и сравнивать ZNCC/Census в некотором фиксированном по размеру окне, т.к. поверхность может быть **под углом**.

Как это исправить если нам известно не только значение диспаритета, но и нормаль поверхности n ?

Какие есть проблемы в SGM?

- 1) В SGM мы предполагаем что все патчи - фрonto-параллельные. А значит как только углы наблюдения далеки от 90 градусов - мы ошибаемся.

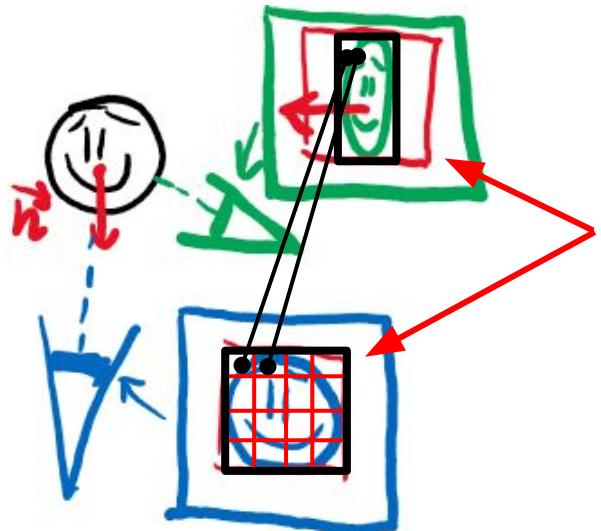


Нельзя просто брать и сравнивать ZNCC/Census в некотором фиксированном по размеру окне, т.к. поверхность может быть **под углом**.

Как это исправить если нам известно не только значение диспаритета, но и нормаль поверхности n ?

Какие есть проблемы в SGM?

- 1) В SGM мы предполагаем что все патчи - фрonto-параллельные. А значит как только углы наблюдения далеки от 90 градусов - мы ошибаемся.

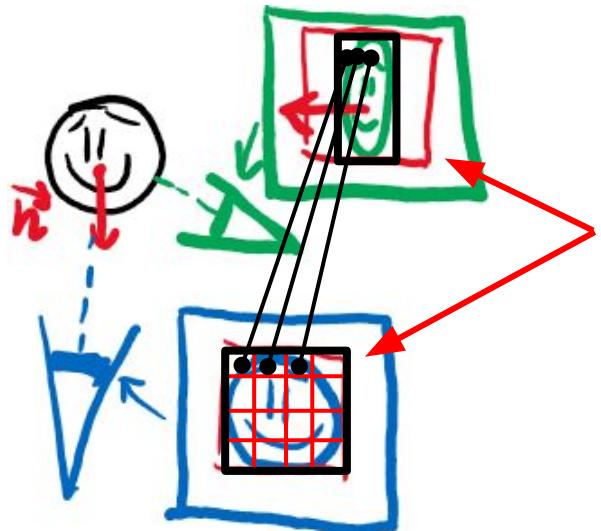


Нельзя просто брать и сравнивать ZNCC/Census в некотором фиксированном по размеру окне, т.к. поверхность может быть **под углом**.

Как это исправить если нам известно не только значение диспаритета, но и нормаль поверхности n ?

Какие есть проблемы в SGM?

- 1) В SGM мы предполагаем что все патчи - фрonto-параллельные. А значит как только углы наблюдения далеки от 90 градусов - мы ошибаемся.

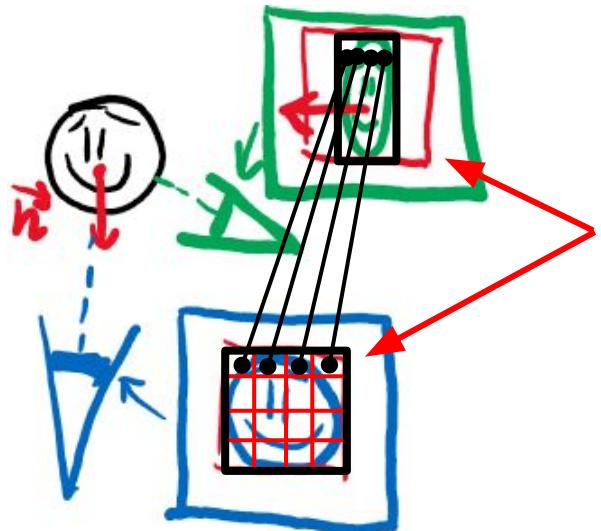


Нельзя просто брать и сравнивать ZNCC/Census в некотором фиксированном по размеру окне, т.к. поверхность может быть **под углом**.

Как это исправить если нам известно не только значение диспаритета, но и нормаль поверхности n ?

Какие есть проблемы в SGM?

- 1) В SGM мы предполагаем что все патчи - фрonto-параллельные. А значит как только углы наблюдения далеки от 90 градусов - мы ошибаемся.

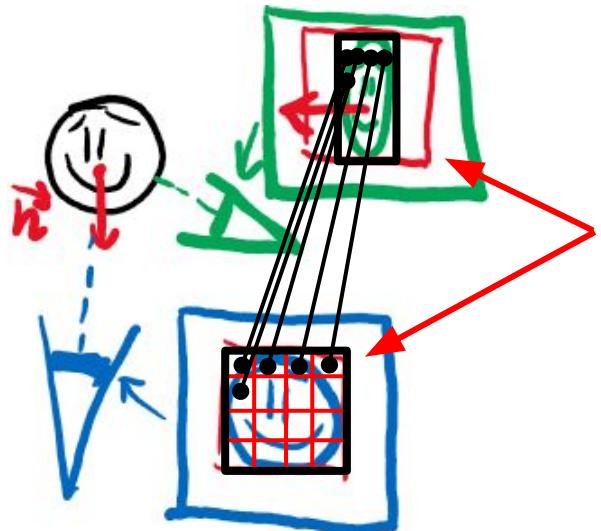


Нельзя просто брать и сравнивать ZNCC/Census в некотором фиксированном по размеру окне, т.к. поверхность может быть **под углом**.

Как это исправить если нам известно не только значение диспаритета, но и нормаль поверхности n ?

Какие есть проблемы в SGM?

- 1) В SGM мы предполагаем что все патчи - фрonto-параллельные. А значит как только углы наблюдения далеки от 90 градусов - мы ошибаемся.



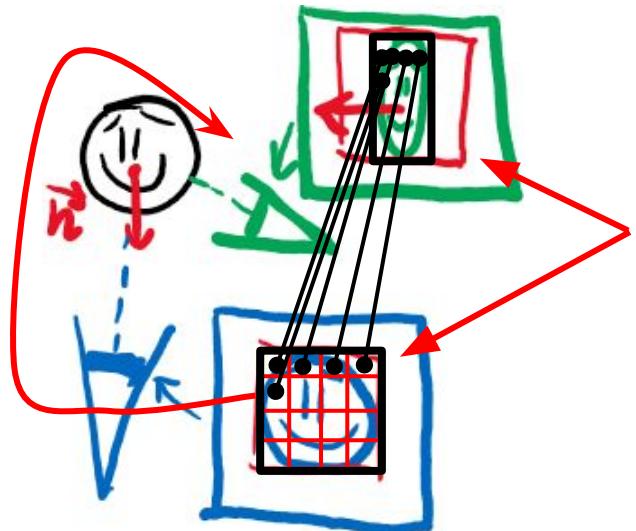
Нельзя просто брать и сравнивать ZNCC/Census в некотором фиксированном по размеру окне, т.к. поверхность может быть **под углом**.

Как это исправить если нам известно не только значение диспаритета, но и нормаль поверхности n ?

Как реализовать такое сопоставление окрестности патча?

Какие есть проблемы в SGM?

- 1) В SGM мы предполагаем что все патчи - фрonto-параллельные. А значит как только углы наблюдения далеки от 90 градусов - мы ошибаемся.



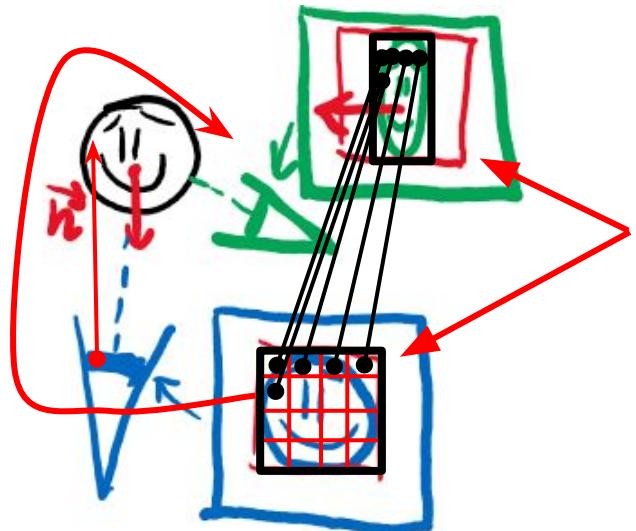
Нельзя просто брать и сравнивать ZNCC/Census в некотором фиксированном по размеру окне, т.к. поверхность может быть **под углом**.

Как это исправить если нам известно не только значение диспаритета, но и нормаль поверхности n ?

Как реализовать такое сопоставление окрестности патча?

Какие есть проблемы в SGM?

- 1) В SGM мы предполагаем что все патчи - фрonto-параллельные. А значит как только углы наблюдения далеки от 90 градусов - мы ошибаемся.

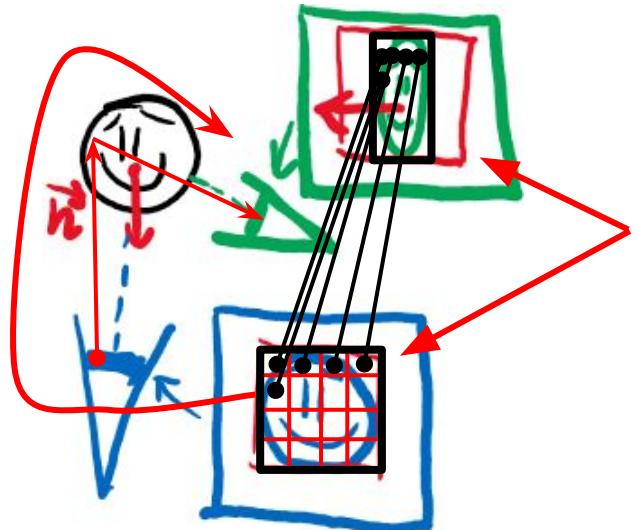


Нельзя просто брать и сравнивать ZNCC/Census в некотором фиксированном по размеру окне, т.к. поверхность может быть **под углом**.

- 1) Пустили в пространстве **луч** соответствующий пикслю и **пересекли** его с плоскостью цели (ведь мы знаем нормаль + значение диспаритета)

Какие есть проблемы в SGM?

- 1) В SGM мы предполагаем что все патчи - фронтопараллельные. А значит как только углы наблюдения далеки от 90 градусов - мы ошибаемся.



Нельзя просто брать и сравнивать ZNCC/Census в некотором фиксированном по размеру окне, т.к. поверхность может быть **под углом**.

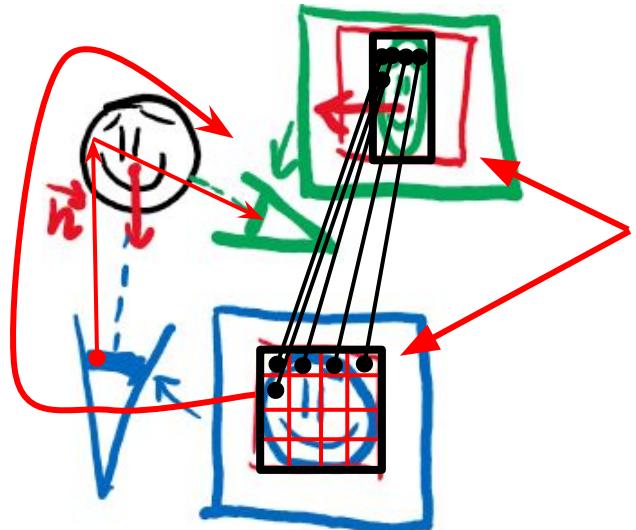
- 1) Пустили в пространстве **луч** соответствующий пикслю и **пересекли** его с плоскостью цели (ведь мы знаем нормаль + значение диспаритета)
- 2) Спроектировали точку поверхности во вторую камеру
- нашли координаты парного пикселя

Какие есть проблемы в SGM?

- 1) В SGM мы предполагаем что все патчи - фрonto-параллельные. А значит как только углы наблюдения далеки от 90 градусов - мы ошибаемся.

ОЧЕНЬ МЕДЛЕННО!

- Плоскость = нормаль + точка, значит нужно пересчитать диспаритет в глубину



Нельзя просто брать и сравнивать ZNCC/Census в некотором фиксированном по размеру окне, т.к. поверхность может быть **под углом**.

- 1) Пустили в пространстве **луч** соответствующий пикслю и **пересекли** его с плоскостью цели (ведь мы знаем нормаль + значение диспаритета)
- 2) Спроектировали точку поверхности во вторую камеру
- нашли координаты парного пикселя

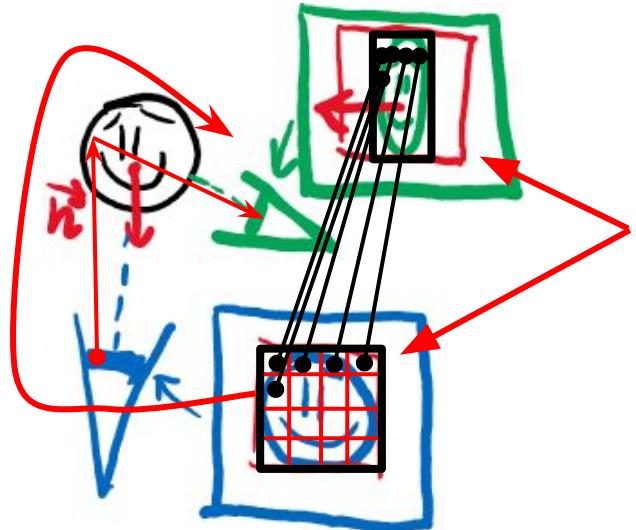
Какие есть проблемы в SGM?

- 1) В SGM мы предполагаем что все патчи - фрonto-параллельные. А значит как только углы наблюдения далеки от 90 градусов - мы ошибаемся.

ОЧЕНЬ МЕДЛЕННО!

- Плоскость = нормаль + точка, значит нужно пересчитать диспаритет в глубину
- Нельзя преподсчитать Census, каждая пара патчей считает свой Cost для каждого диспаритета

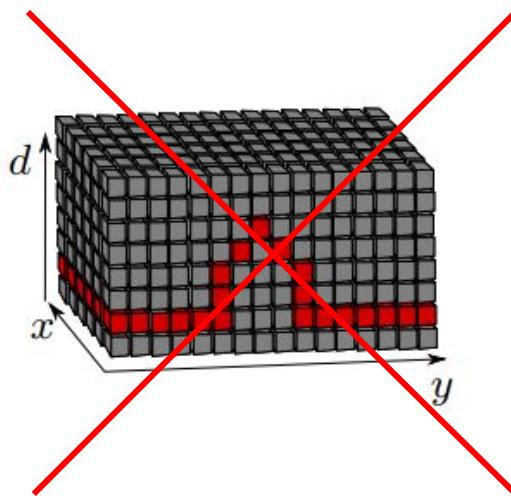
Нельзя просто брать и сравнивать ZNCC/Census в некотором фиксированном по размеру окне, т.к. поверхность может быть **под углом**.



- 1) Пустили в пространстве **луч** соответствующий пикслю и **пересекли** его с плоскостью цели (ведь мы знаем нормаль + значение диспаритета)
- 2) Спроектировали точку поверхности во вторую камеру
 - нашли координаты парного пикселя

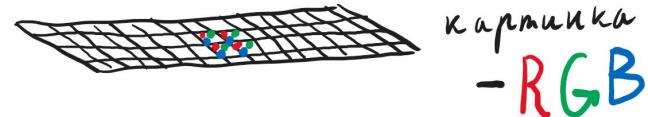
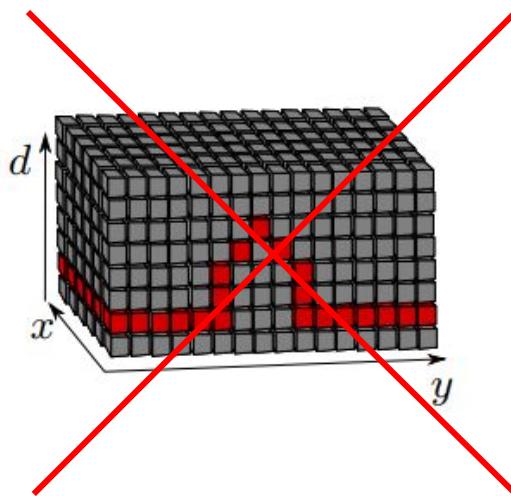
Какие есть проблемы в **SGM**?

- 1) В **SGM** мы предполагаем что все патчи - фронтопараллельные.
А теперь?



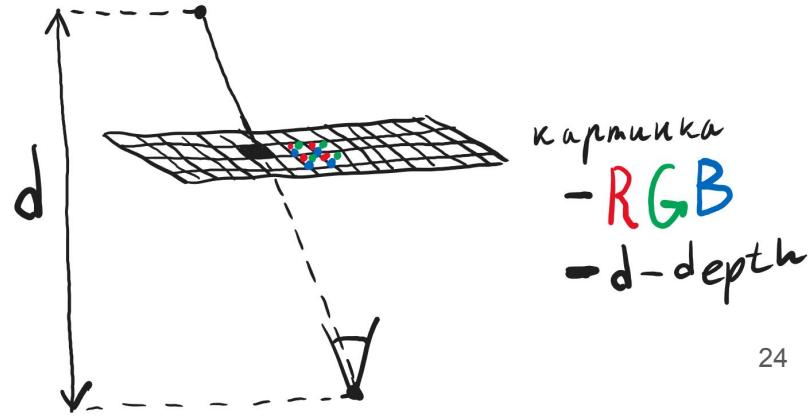
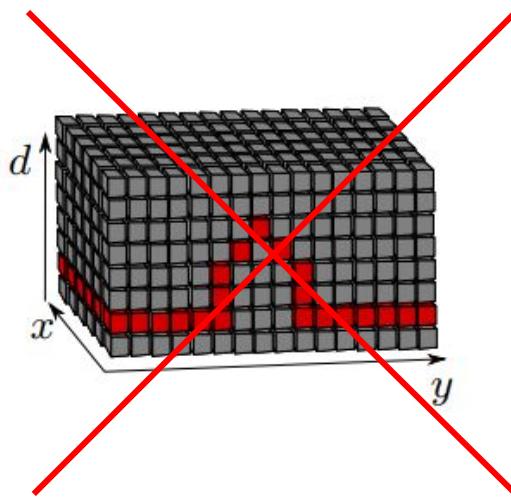
Какие есть проблемы в SGM?

- 1) В SGM мы предполагаем что все патчи - фронтопараллельные.
А теперь?



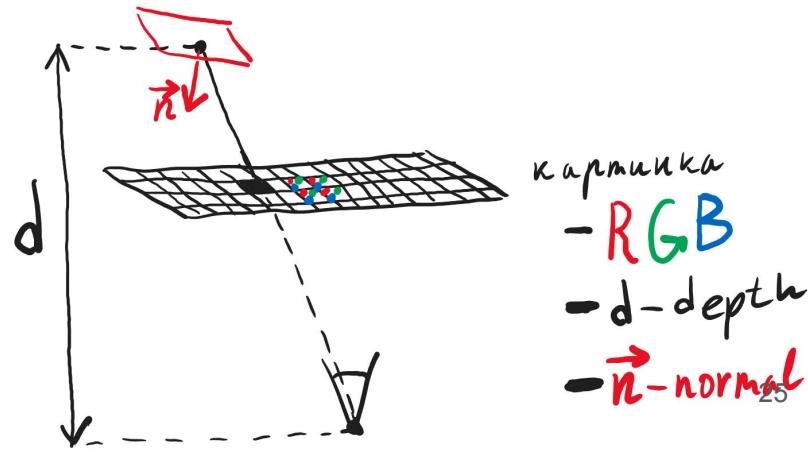
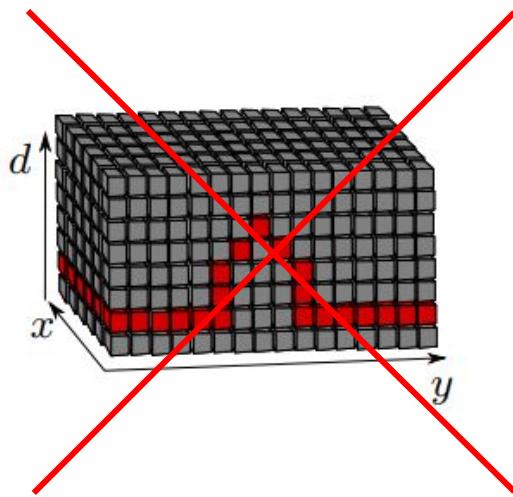
Какие есть проблемы в SGM?

- 1) В SGM мы предполагаем что все патчи - фрonto-параллельные.
Храним 3D положение (**глубина**).



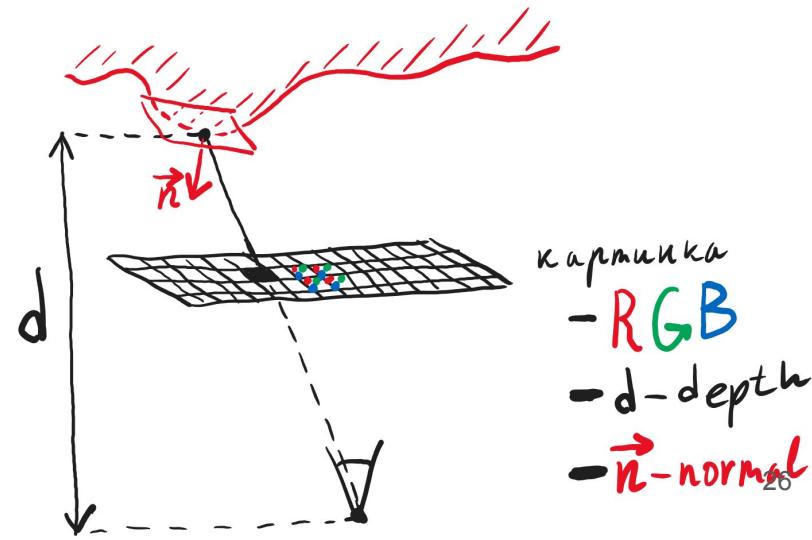
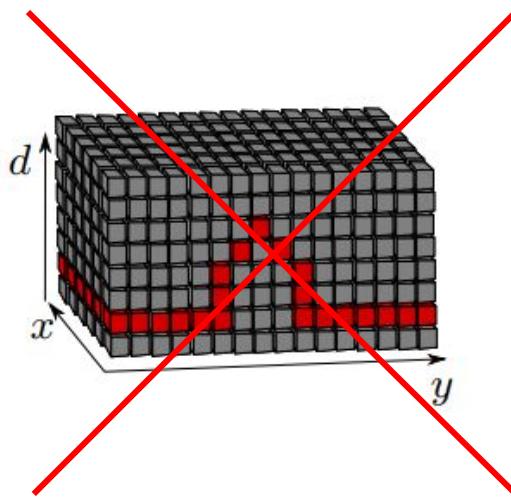
Какие есть проблемы в SGM?

- 1) В SGM мы предполагаем что все патчи - фрonto-параллельные.
В дополнение к 3D положению (**глубина**) - храним в пикселе **нормаль**.



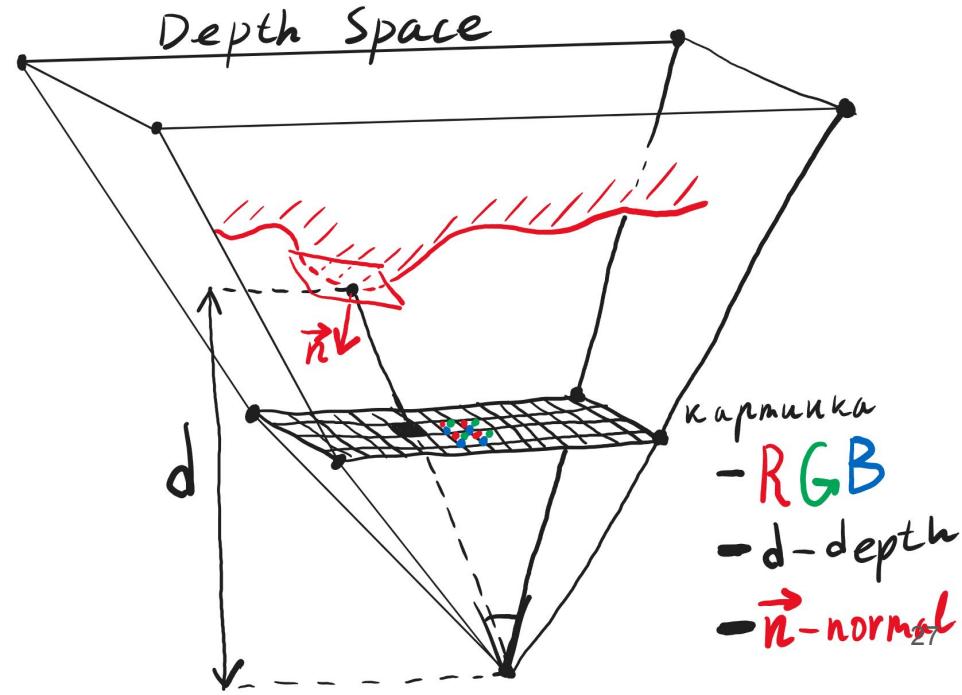
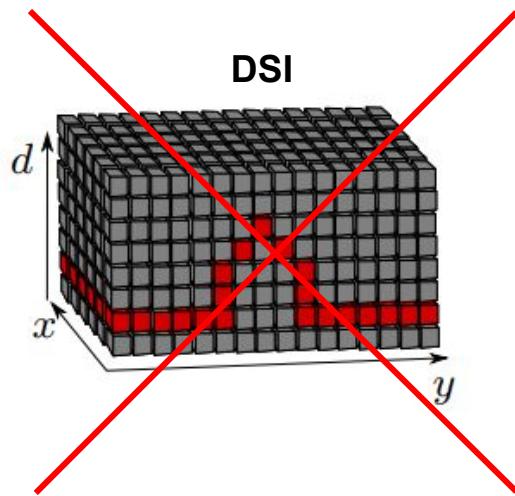
Какие есть проблемы в SGM?

- 1) В SGM мы предполагаем что все патчи - фрonto-параллельные.
В дополнение к 3D положению (**глубина**) - храним в пикселе **нормаль**.



Какие есть проблемы в SGM?

- 1) В SGM мы предполагаем что все патчи - фрonto-параллельные.
В дополнение к 3D положению (**глубина**) - храним в пикселе **нормаль**.



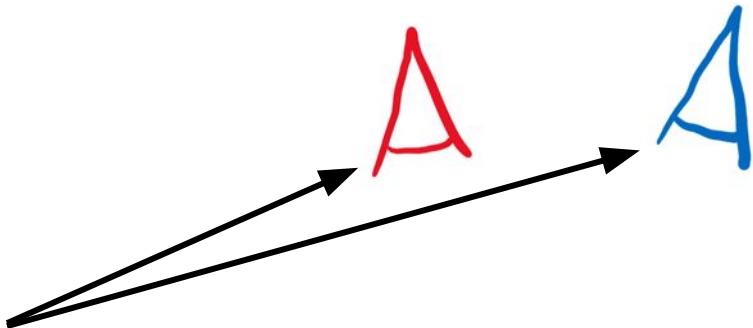
Какие есть проблемы в **SGM**?

- 1) В **SGM** мы предполагаем что все патчи - фронтопараллельные.
В дополнение к 3D положению (**глубина**) - храним в пикселе **нормаль**.
- 2) Что делать с бликами? Есть ли улики которые мы упускаем?

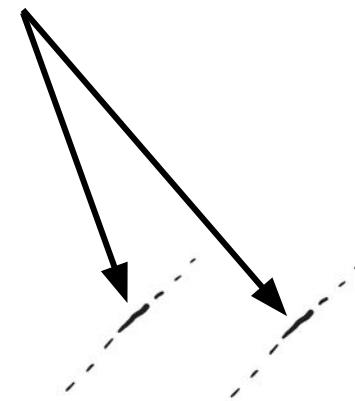


Какие есть проблемы в **SGM**?

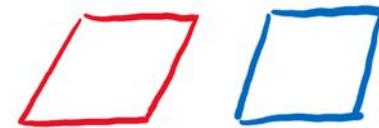
- 1) В **SGM** мы предполагаем что все патчи - фронтопараллельные.
В дополнение к 3D положению (**глубина**) - храним в пикселе **нормаль**.
- 2) Что делать с бликами? Есть ли улики которые мы упускаем?



Две камеры смотрят вниз
на два горизонтальных провода

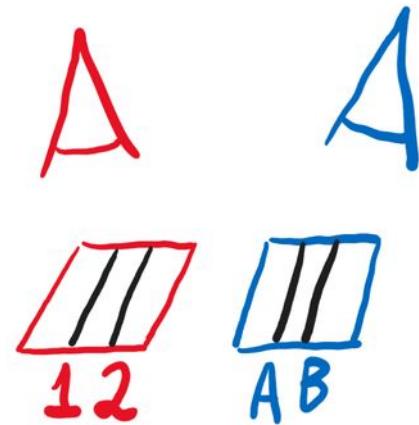


A A

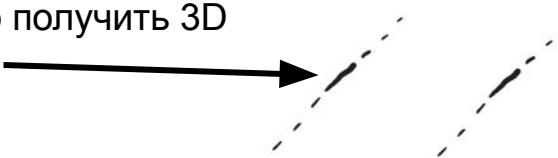


A A

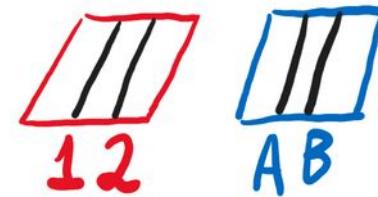




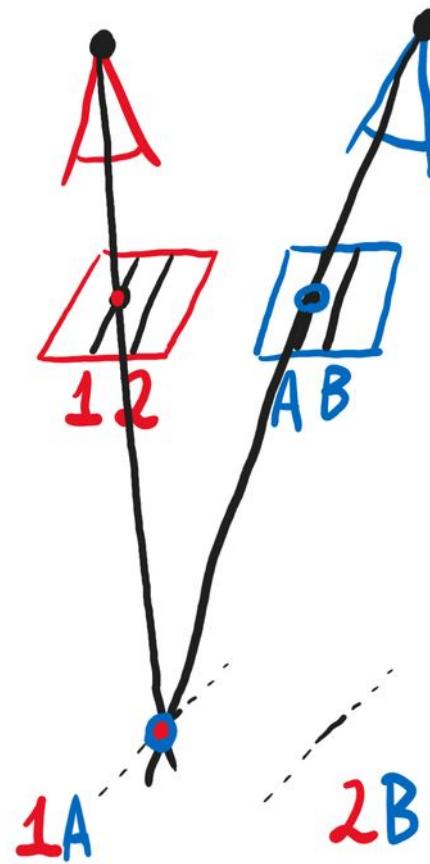
Триангуляцией каких проецированных в фотографию проводов можно получить 3D координаты левого провода?

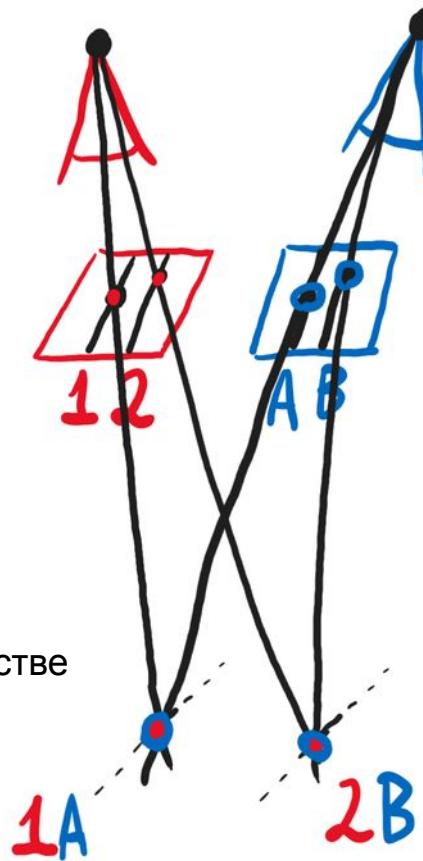


A A

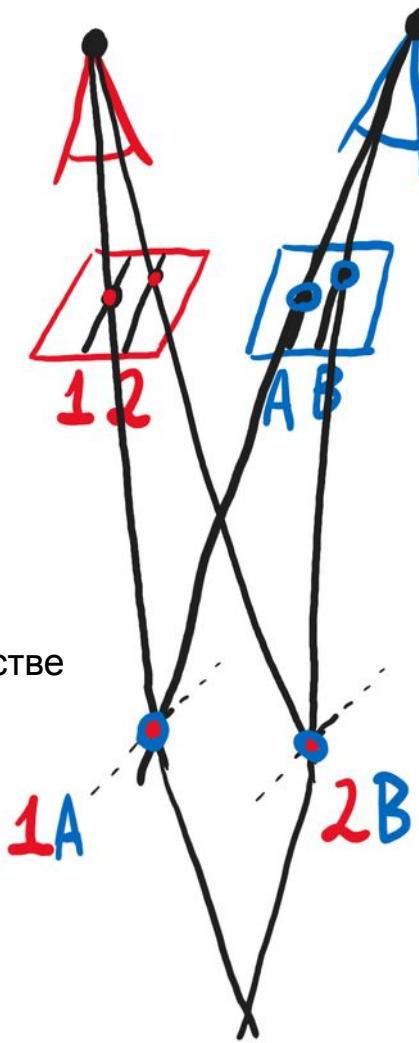


1A 2B

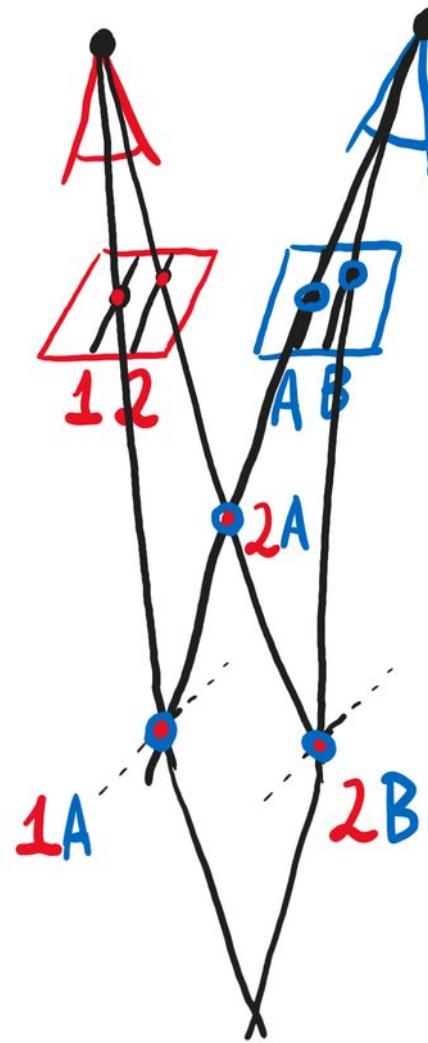


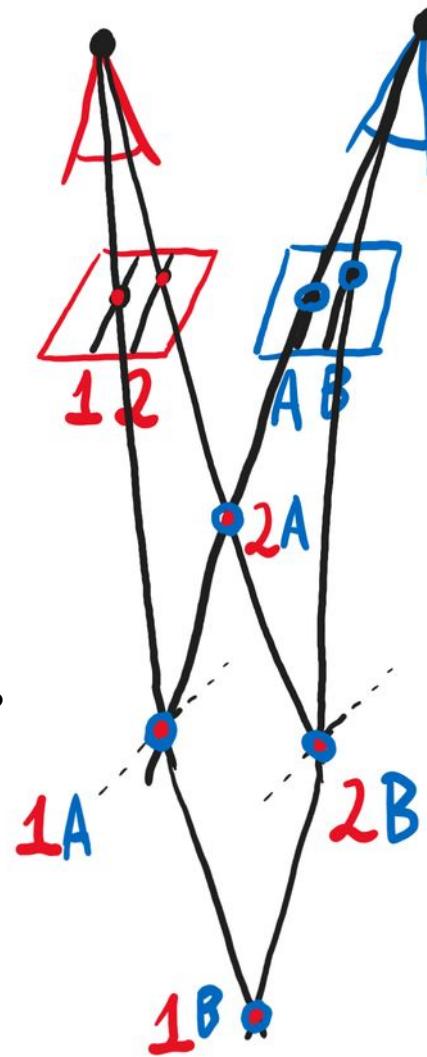


Если у нас есть две такие фотографии.
Уверены ли мы что провода в пространстве
расположены именно так?

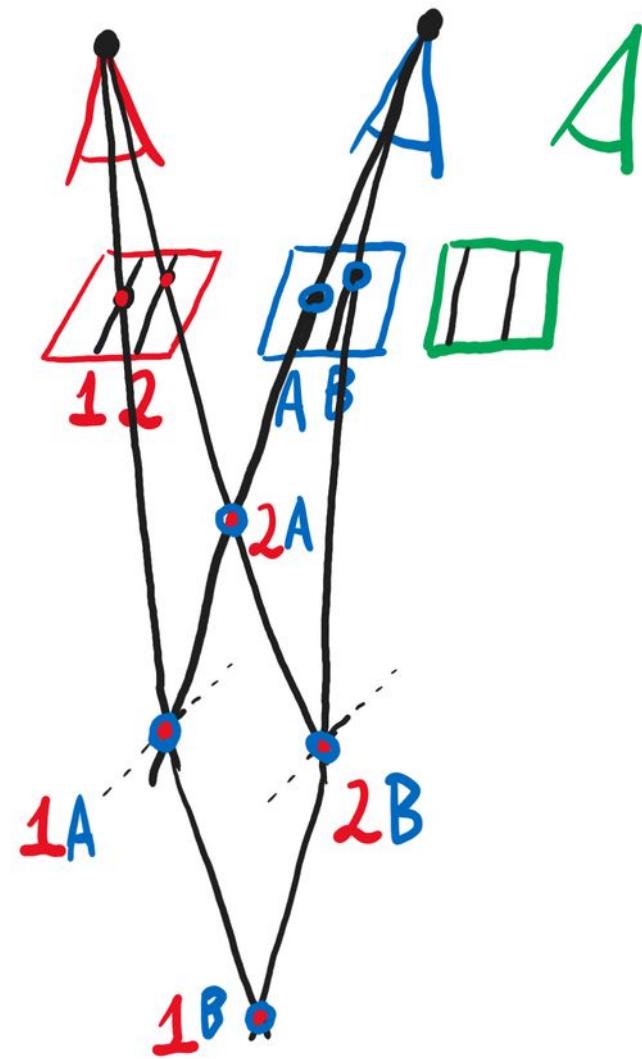


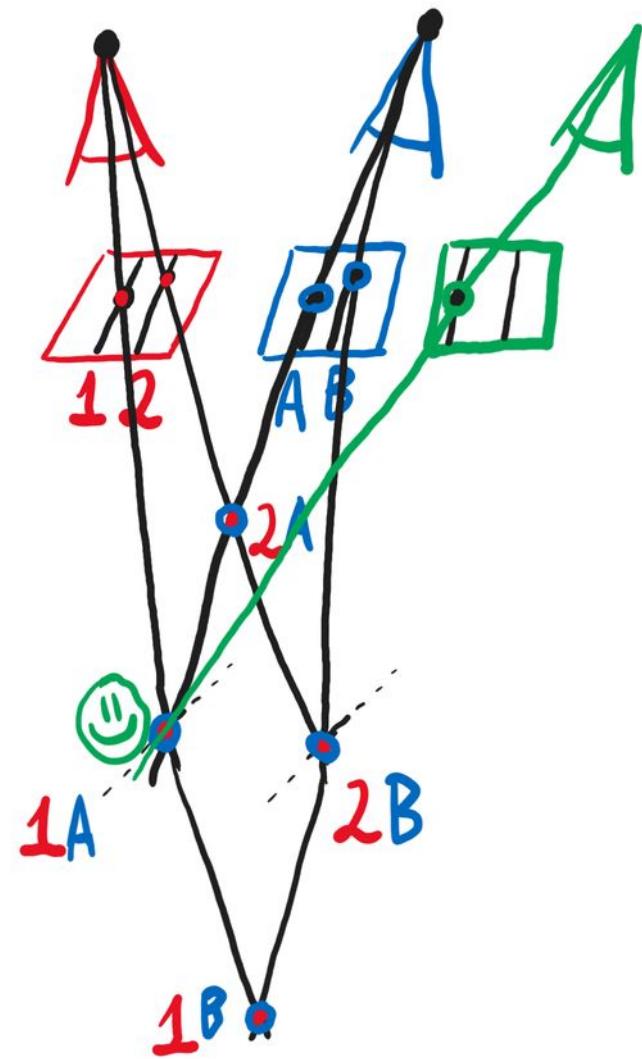
Если у нас есть две такие фотографии.
Уверены ли мы что провода в пространстве
расположены именно так?

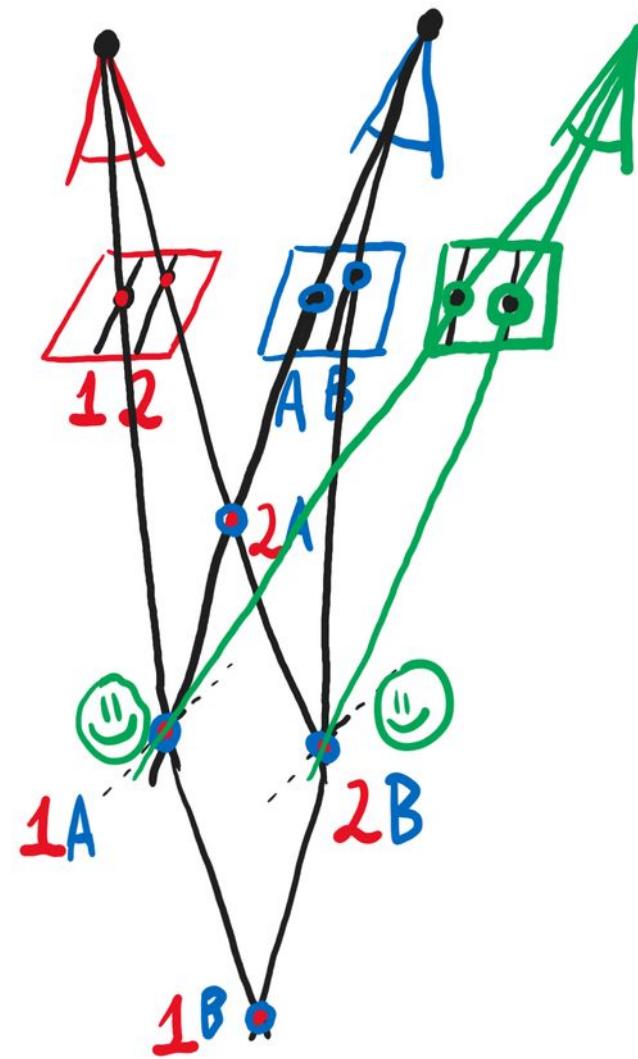


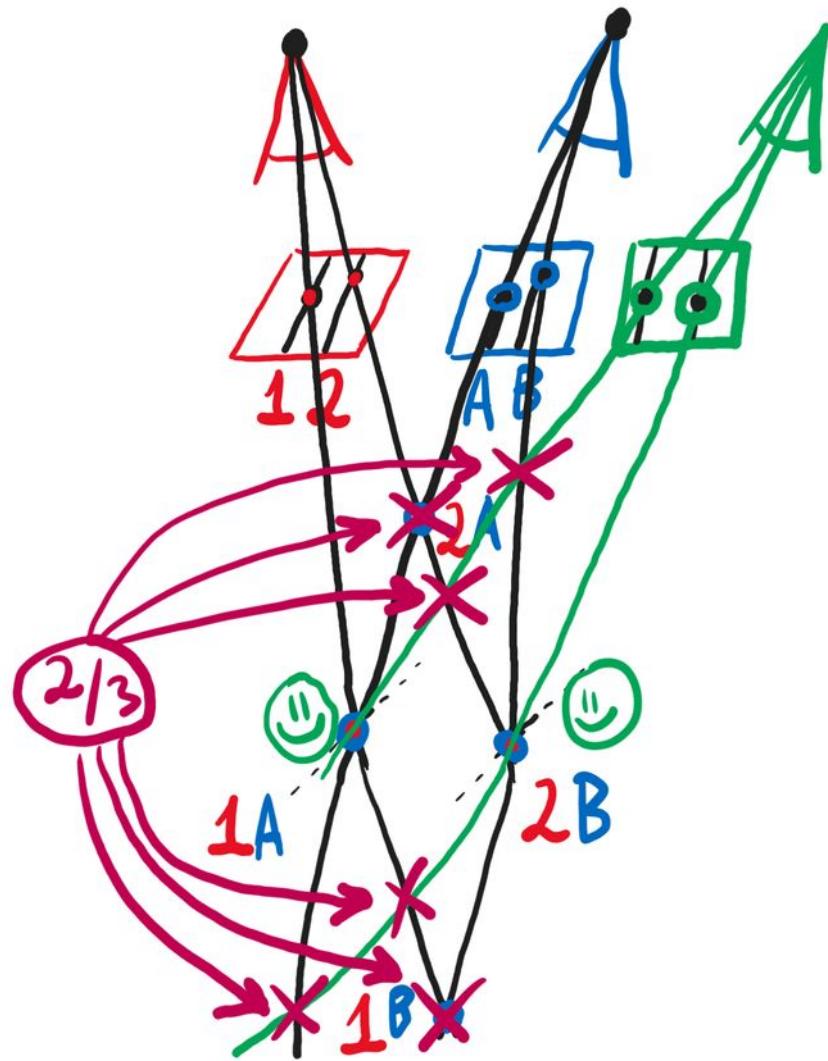


Как разрешить эту неоднозначность?
Что бы вы сделали если бы у вас была
возможность отправить дрон повторно?









Какие есть проблемы в **SGM**?

- 1) В **SGM** мы предполагаем что все патчи - фронтопараллельные.
В дополнение к 3D положению (диспаритет) - храним в пикселе нормаль.
- 2) **Что делать с бликами? Есть ли улики которые мы упускаем?**

Какие есть проблемы в **SGM**?

- 1) В **SGM** мы предполагаем что все патчи - фронтопараллельные.
В дополнение к 3D положению (диспаритет) - храним в пикселе нормаль.
- 2) Не учитывали остальные камеры - только одну стереопару.
Учитывать все камеры - добавить робастности (устойчивости).

Какие есть проблемы в **SGM**?

- 1) В **SGM** мы предполагаем что все патчи - фронтопараллельные.
В дополнение к 3D положению (диспаритет) - храним в пикселе нормаль.
- 2) Не учитывали остальные камеры - только одну стереопару.
Учитывать все камеры - добавить робастности (устойчивости).
И поможет не ошибаться в случае повторяющихся/регулярных структур.

Какие есть проблемы в **SGM**?

- 1) В **SGM** мы предполагаем что все патчи - фронтопараллельные.
В дополнение к 3D положению (диспаритет) - храним в пикселе нормаль.
- 2) Не учитывали остальные камеры - только одну стереопару.
Учитывать все камеры - добавить робастности (устойчивости).
И поможет не ошибаться в случае повторяющихся/регулярных структур.
- 3) **От чего зависит качество тонких структур?** (тонкие перила, шпили, ветки)

Какие есть проблемы в **SGM**?

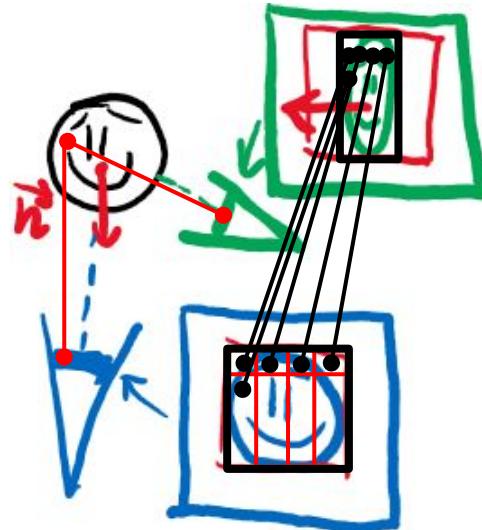
- 1) В **SGM** мы предполагаем что все патчи - фронтопараллельные.
В дополнение к 3D положению (диспаритет) - храним в пикселе нормаль.
- 2) Не учитывали остальные камеры - только одну стереопару.
Учитывать все камеры - добавить робастности (устойчивости).
И поможет не ошибаться в случае повторяющихся/регулярных структур.
- 3) Качество тонких структур (тонкие перила, шпили, ветки) зависит от размера патча.
Что нас сдерживает от его уменьшения скажем до 5×5 или до 3×3 ?

Какие есть проблемы в **SGM**?

- 1) В **SGM** мы предполагаем что все патчи - фронтопараллельные.
В дополнение к 3D положению (диспаритет) - храним в пикселе нормаль.
- 2) Не учитывали остальные камеры - только одну стереопару.
Учитывать все камеры - добавить робастности (устойчивости).
И поможет не ошибаться в случае повторяющихся/регулярных структур.
- 3) Качество тонких структур (тонкие перила, шпили, ветки) зависит от размера патча.
Добавление информации из остальных камер добавит робастности и однозначности - настолько, что можно будет использовать патч **5x5/3x3**.

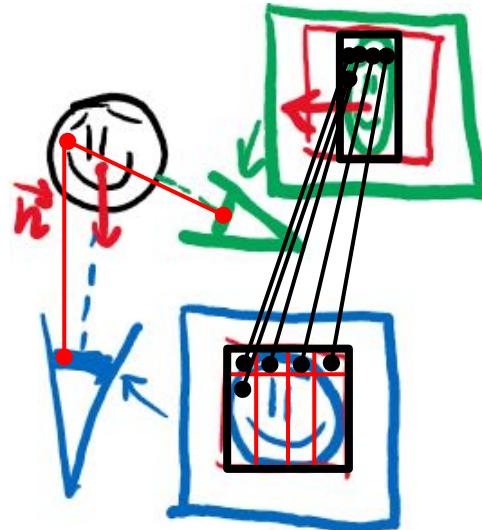
Итого: идеи/точки роста качества

- 1) В каждом пикселе храним еще и нормаль - чтобы учитывать перспективное искажение и измерять похожесть патчей точнее.



Итого: идеи/точки роста качества

- 1) В каждом пикселе храним еще и нормаль - чтобы учитывать перспективное искажение и измерять похожесть патчей точнее.



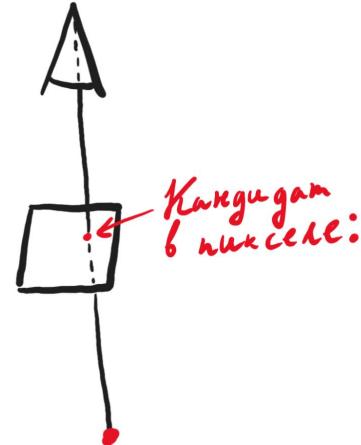
Итого: идеи/точки роста качества

- 1) В каждом пикселе храним еще и нормаль - чтобы учитывать перспективное искажение и измерять похожесть патчей точнее.
- 2) Похожесть патча меряем не только со второй камерой - но и со всеми остальными.



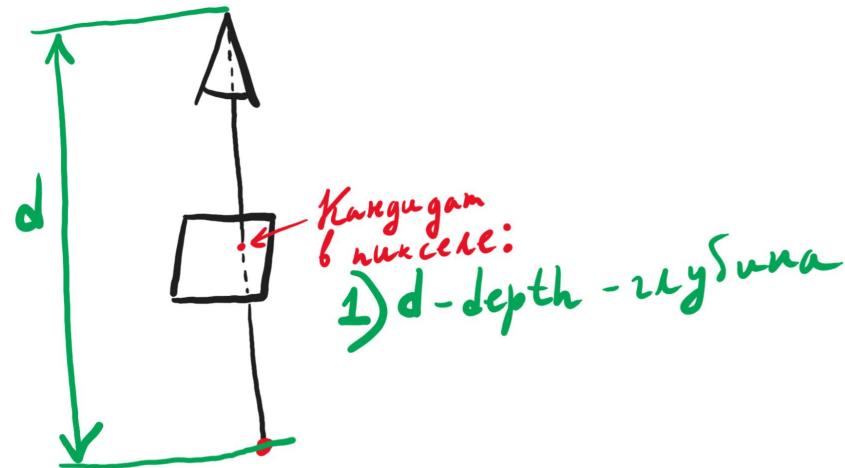
Итого: идеи/точки роста качества

- 1) В каждом пикселе храним еще и нормаль - чтобы учитывать перспективное искажение и измерять похожесть патчей точнее.
- 2) Похожесть патча меряем не только со второй камерой - но и со всеми остальными.



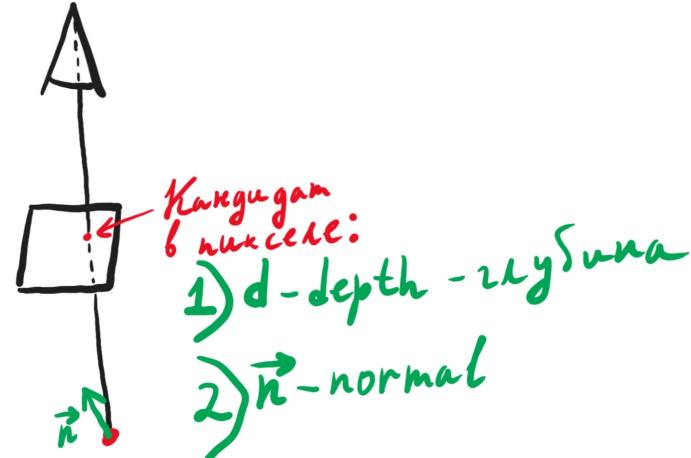
Итого: идеи/точки роста качества

- 1) В каждом пикселе храним еще и нормаль - чтобы учитывать перспективное искажение и измерять похожесть патчей точнее.
- 2) Похожесть патча меряем не только со второй камерой - но и со всеми остальными.



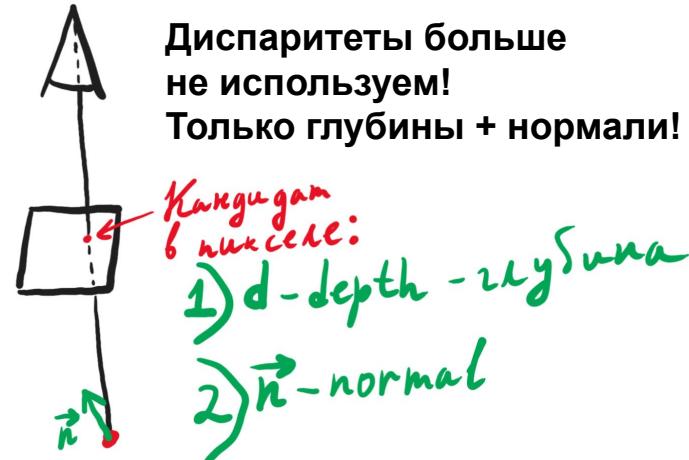
Итого: идеи/точки роста качества

- 1) В каждом пикселе храним еще и нормаль - чтобы учитывать перспективное искажение и измерять похожесть патчей точнее.
- 2) Похожесть патча меряем не только со второй камерой - но и со всеми остальными.



Итого: идеи/точки роста качества

- 1) В каждом пикселе храним еще и нормаль - чтобы учитывать перспективное искажение и измерять похожесть патчей точнее.
- 2) Похожесть патча меряем не только со второй камерой - но и со всеми остальными.



Итого: идеи/точки роста качества

- 1) В каждом пикселе храним еще и нормаль - чтобы учитывать перспективное искажение и измерять похожесть патчей точнее.
- 2) Похожесть патча меряем не только со второй камерой - но и со всеми остальными.



Итого: идеи/точки роста качества

- 1) В каждом пикселе храним еще и нормаль - чтобы учитывать перспективное искажение и измерять похожесть патчей точнее.
- 2) Похожесть патча меряем не только со второй камерой - но и со всеми остальными.



Плоскость поверхности вокруг патча порождена гипотезой пикселя:

- центральной точкой (она определена глубиной d)

Как идеально вычисляются ее координаты?

Итого: идеи/точки роста качества

- 1) В каждом пикселе храним еще и нормаль - чтобы учитывать перспективное искажение и измерять похожесть патчей точнее.
- 2) Похожесть патча меряем не только со второй камерой - но и со всеми остальными.



Плоскость поверхности вокруг патча порождена гипотезой пикселя:

- центральной точкой (она определена глубиной d)

Как идеально вычисляются ее координаты?

$(x, y, z) = \text{camera.unproject(pixel_i, pixel_j, d)}$

Итого: идеи/точки роста качества

- 1) В каждом пикселе храним еще и нормаль - чтобы учитывать перспективное искажение и измерять похожесть патчей точнее.
- 2) Похожесть патча меряем не только со второй камерой - но и со всеми остальными.



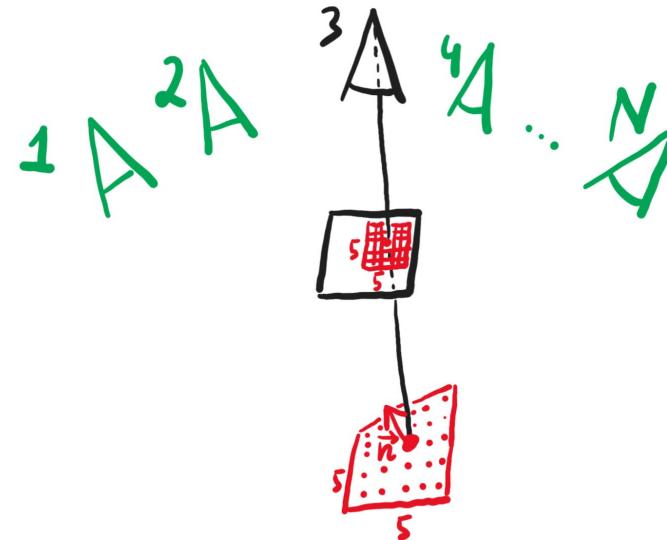
Плоскость поверхности вокруг патча порождена гипотезой пикселя:

- центральной точкой (она определена глубиной d)
- нормалью к поверхности n (дана вместе с d)

Т.е. по сути в каждом пикселе мы хотим плоскость-гипотезу

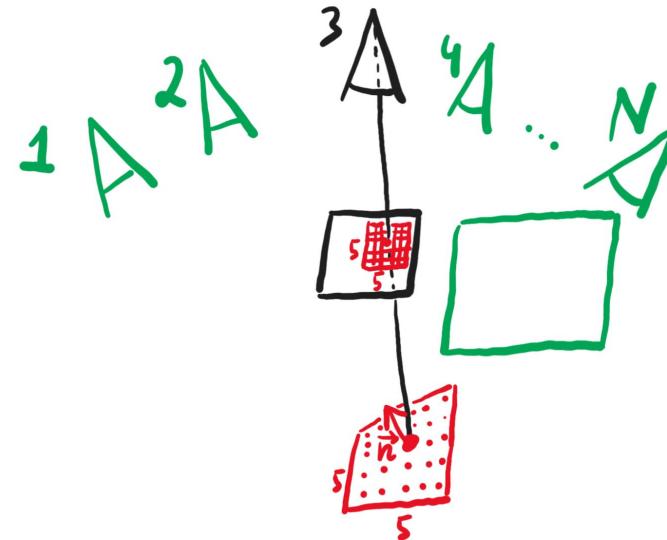
Итого: идеи/точки роста качества

- 1) В каждом пикселе храним еще и нормаль - чтобы учитывать перспективное искажение и измерять похожесть патчей точнее.
- 2) Похожесть патча меряем не только со второй камерой - но и со всеми остальными.



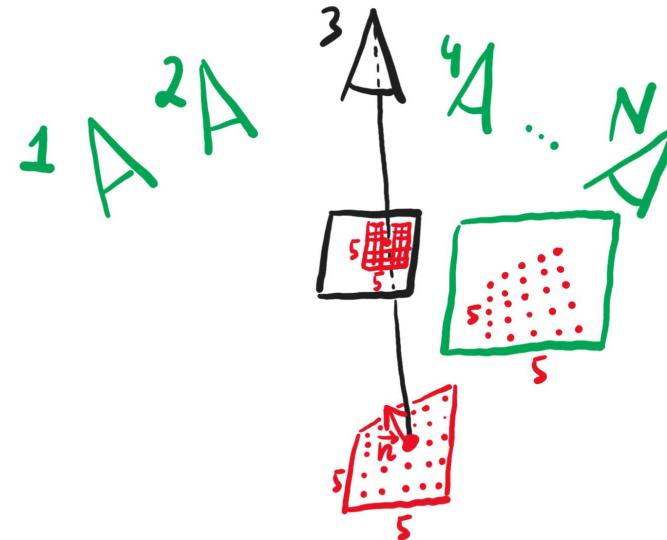
Итого: идеи/точки роста качества

- 1) В каждом пикселе храним еще и нормаль - чтобы учитывать перспективное искажение и измерять похожесть патчей точнее.
- 2) Похожесть патча меряем не только со второй камерой - но и со всеми остальными.



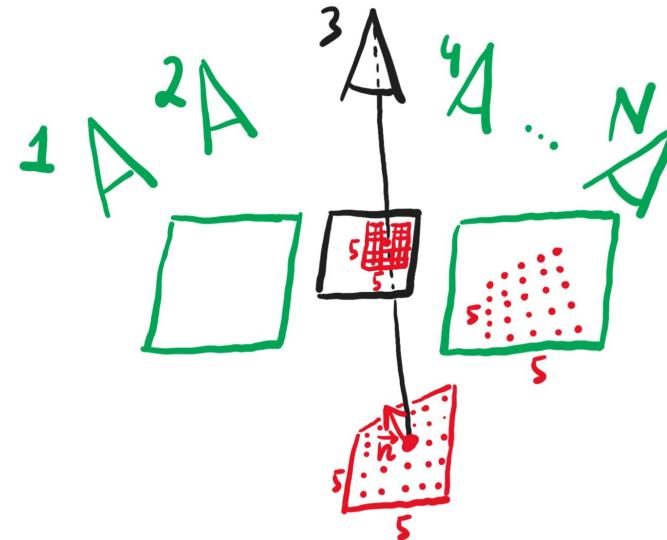
Итого: идеи/точки роста качества

- 1) В каждом пикселе храним еще и нормаль - чтобы учитывать перспективное искажение и измерять похожесть патчей точнее.
- 2) Похожесть патча меряем не только со второй камерой - но и со всеми остальными.



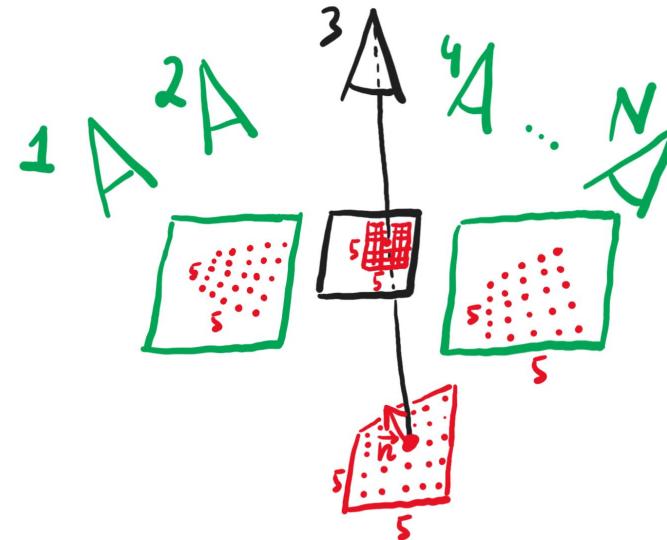
Итого: идеи/точки роста качества

- 1) В каждом пикселе храним еще и нормаль - чтобы учитывать перспективное искажение и измерять похожесть патчей точнее.
- 2) Похожесть патча меряем не только со второй камерой - но и со всеми остальными.



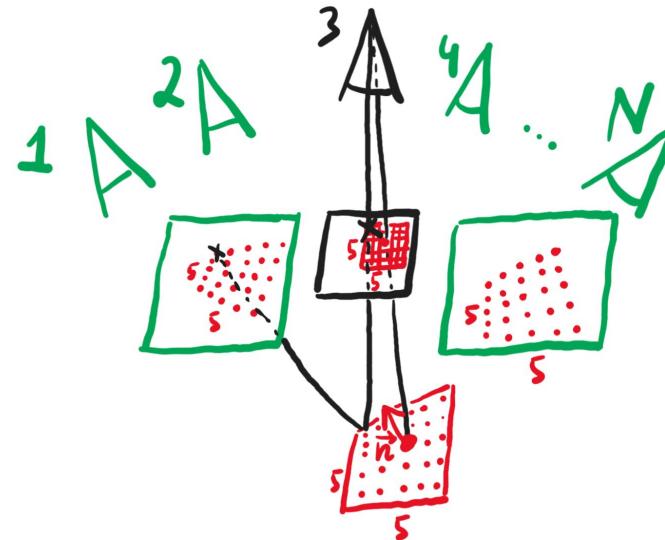
Итого: идеи/точки роста качества

- 1) В каждом пикселе храним еще и нормаль - чтобы учитывать перспективное искажение и измерять похожесть патчей точнее.
- 2) Похожесть патча меряем не только со второй камерой - но и со всеми остальными.



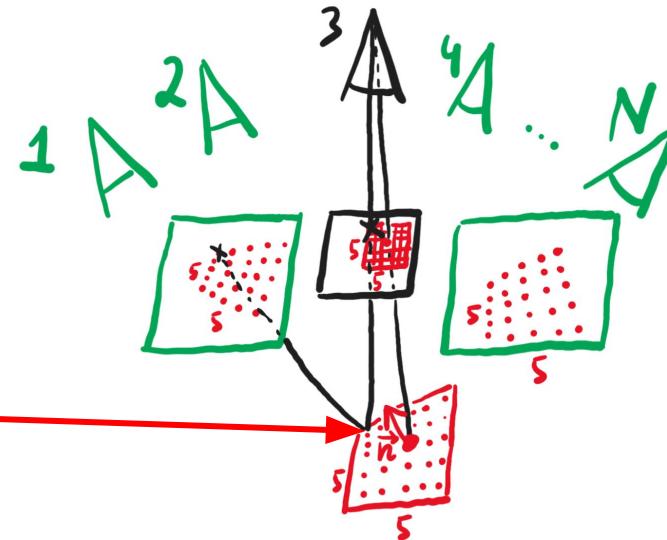
Итого: идеи/точки роста качества

- 1) В каждом пикселе храним еще и нормаль - чтобы учитывать перспективное искажение и измерять похожесть патчей точнее.
- 2) Похожесть патча меряем не только со второй камерой - но и со всеми остальными.



Итого: идеи/точки роста качества

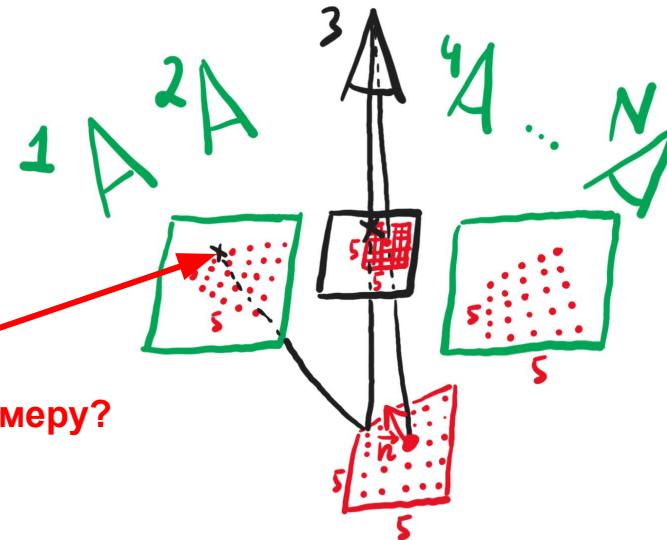
- 1) В каждом пикселе храним еще и нормаль - чтобы учитывать перспективное искажение и измерять похожесть патчей точнее.
- 2) Похожесть патча меряем не только со второй камерой - но и со всеми остальными.



Как найти 3D координаты этой точки?

Итого: идеи/точки роста качества

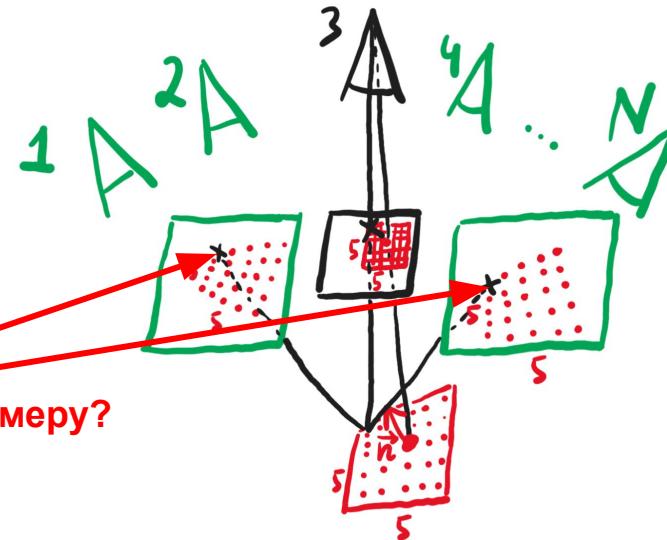
- 1) В каждом пикселе храним еще и нормаль - чтобы учитывать перспективное искажение и измерять похожесть патчей точнее.
- 2) Похожесть патча меряем не только со второй камерой - но и со всеми остальными.



Как найти координаты проекции в другую камеру?

Итого: идеи/точки роста качества

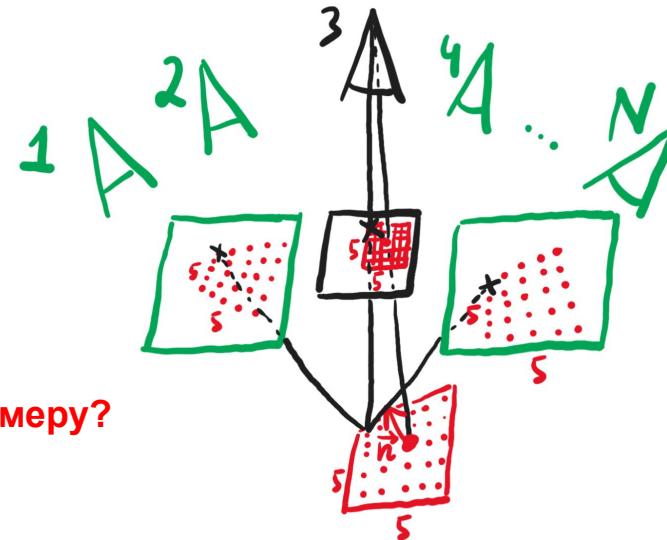
- 1) В каждом пикселе храним еще и нормаль - чтобы учитывать перспективное искажение и измерять похожесть патчей точнее.
- 2) Похожесть патча меряем не только со второй камерой - но и со всеми остальными.



Как найти координаты проекции в другую камеру?

Итого: идеи/точки роста качества

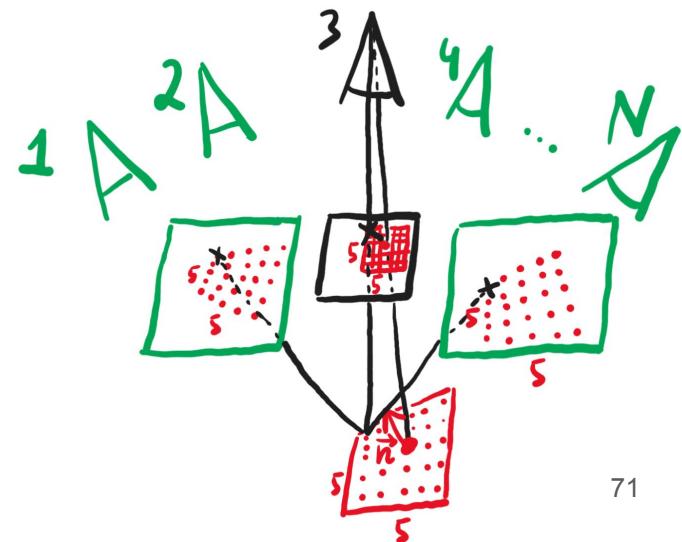
- 1) В каждом пикселе храним еще и нормаль - чтобы учитывать перспективное искажение и измерять похожесть патчей точнее.
- 2) Похожесть патча меряем не только со второй камерой - но и со всеми остальными.



Как найти координаты проекции в другую камеру?
 $(pixel_i, pixel_j) = camera.project(x, y, z)$

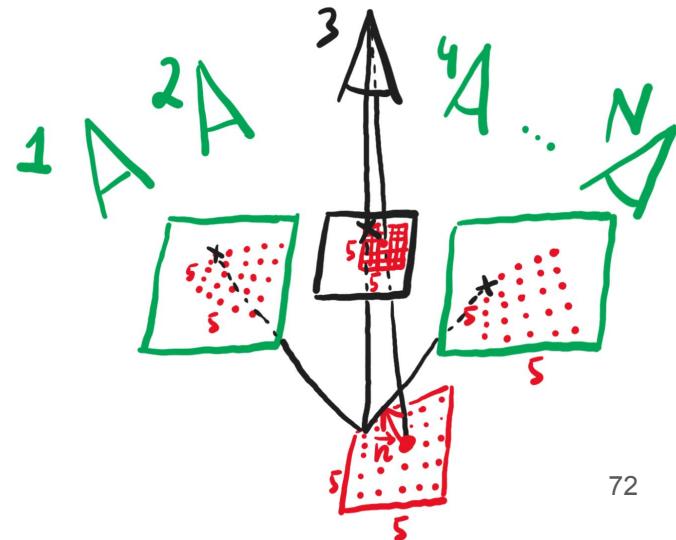
Итого: идеи/точки роста качества

- 1) В каждом пикселе храним еще и нормаль - чтобы учитывать перспективное искажение и измерять похожесть патчей точнее.
- 2) Похожесть патча меряем не только со второй камерой - но и со всеми остальными.
- 3) Будем использовать маленькие патчи ($3 \times 3 / 5 \times 5$) ради тонких структур (балки, перила, шпили).



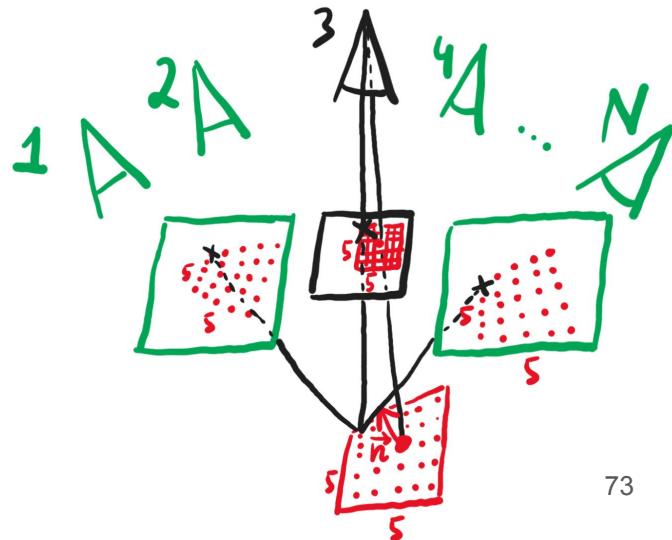
Итого: идеи/точки роста качества

- 1) В каждом пикселе храним еще и нормаль - чтобы учитывать перспективное искажение и измерять похожесть патчей точнее.
- 2) Похожесть патча меряем не только со второй камерой - но и со всеми остальными.
- 3) Будем использовать маленькие патчи ($3 \times 3 / 5 \times 5$) ради тонких структур (балки, перила, шпили).
- 4) Как добавить в $\text{cost}(d, n)$ учет нескольких камер-соседей?



Итого: идеи/точки роста качества

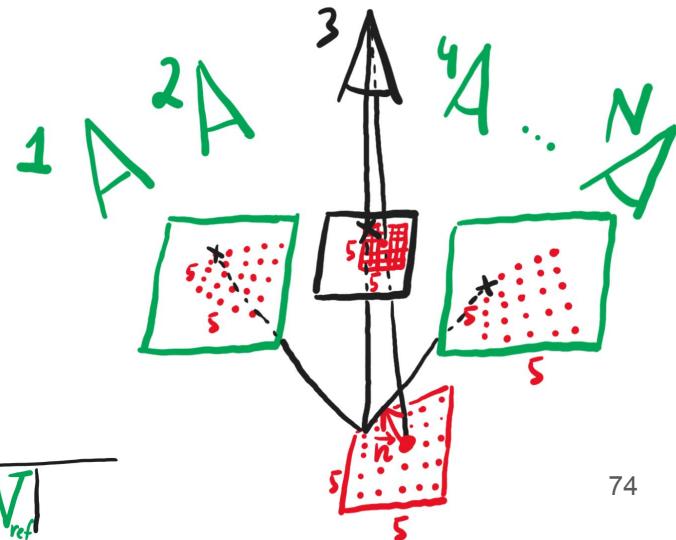
- 1) В каждом пикселе храним еще и нормаль - чтобы учитывать перспективное искажение и измерять похожесть патчей точнее.
- 2) Похожесть патча меряем не только со второй камерой - но и со всеми остальными.
- 3) Будем использовать маленькие патчи (**3x3 / 5x5**) ради тонких структур (балки, перила, шпили).
- 4) Как добавить в **cost(d, n)** учет нескольких камер-соседей? $\text{cost}_{\text{ref}}(i, j, d, \vec{n}) = ???$



Итого: идеи/точки роста качества

- 1) В каждом пикселе храним еще и нормаль - чтобы учитывать перспективное искажение и измерять похожесть патчей точнее.
- 2) Похожесть патча меряем не только со второй камерой - но и со всеми остальными.
- 3) Будем использовать маленькие патчи (**3x3 / 5x5**) ради тонких структур (балки, перила, шпили).
- 4) Как добавить в **cost(d, n)** учет нескольких камер-соседей?

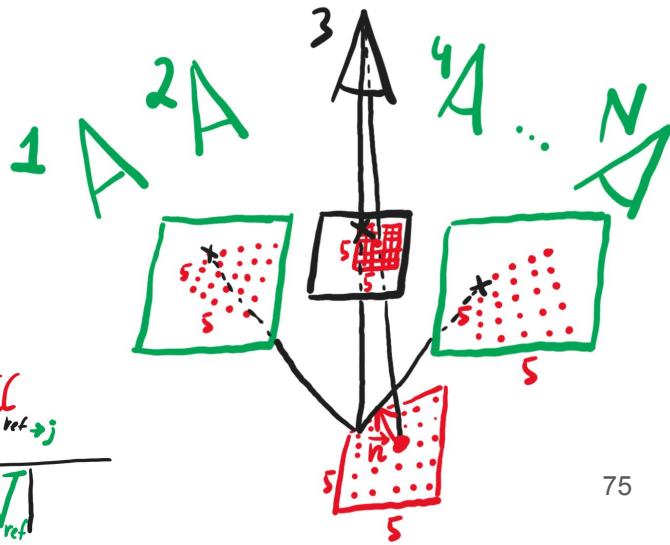
$$cost_{ref}(i, j, d, \vec{n}) = \frac{\sum_{j \in N_{ref}}}{|N_{ref}|}$$



Итого: идеи/точки роста качества

- 1) В каждом пикселе храним еще и нормаль - чтобы учитывать перспективное искажение и измерять похожесть патчей точнее.
- 2) Похожесть патча меряем не только со второй камерой - но и со всеми остальными.
- 3) Будем использовать маленькие патчи (**3x3 / 5x5**) ради тонких структур (балки, перила, шпили).
- 4) Как добавить в $\text{cost}(d, n)$ учет нескольких камер-соседей?

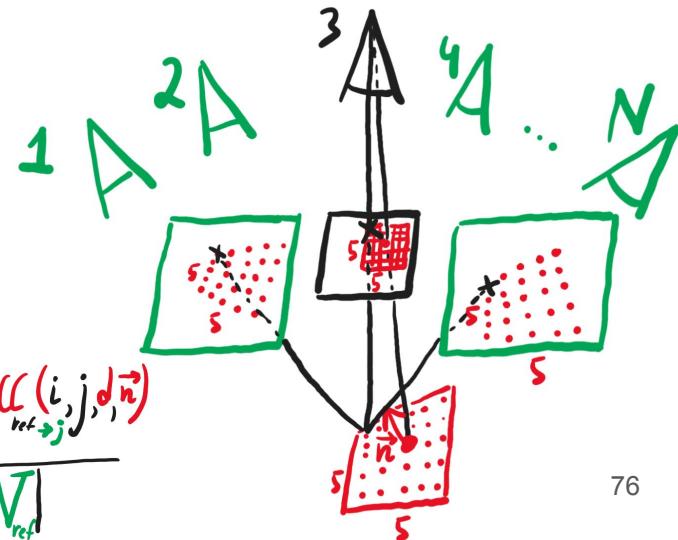
$$\text{cost}_{\text{ref}}(i, j, d, \vec{n}) = \frac{\sum_{j \in N_{\text{ref}}} \text{ZNCC}_{\text{ref} \rightarrow j}}{|N_{\text{ref}}|}$$



Итого: идеи/точки роста качества

- 1) В каждом пикселе храним еще и нормаль - чтобы учитывать перспективное искажение и измерять похожесть патчей точнее.
- 2) Похожесть патча меряем не только со второй камерой - но и со всеми остальными.
- 3) Будем использовать маленькие патчи (**3x3 / 5x5**) ради тонких структур (балки, перила, шпили).
- 4) Как добавить в $\text{cost}(d, n)$ учет нескольких камер-соседей?

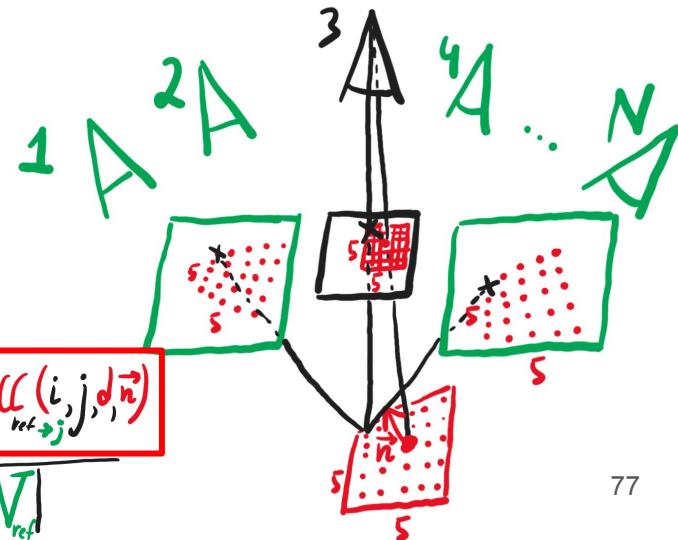
$$\text{cost}_{\text{ref}}(i, j, d, \vec{n}) = \frac{\sum_{j \in N_{\text{ref}}} \text{ZNCC}(i, j, d, \vec{n})}{|N_{\text{ref}}|}$$



Итого: идеи/точки роста качества

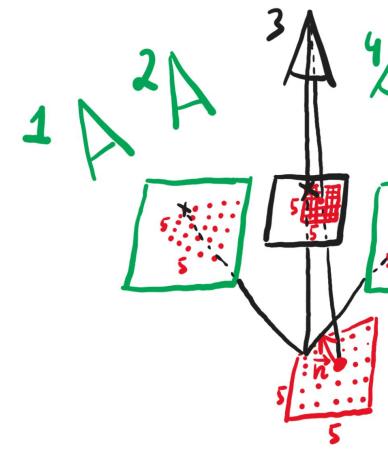
- 1) В каждом пикселе храним еще и нормаль - чтобы учитывать перспективное искажение и измерять похожесть патчей точнее.
- 2) Похожесть патча меряем не только со второй камерой - но и со всеми остальными.
- 3) Будем использовать маленькие патчи (**3x3 / 5x5**) ради тонких структур (балки, перила, шпили).
- 4) Как добавить в $\text{cost}(d, n)$ учет нескольких камер-соседей?

$$\text{cost}_{\text{ref}}(i, j, d, \vec{n}) = \frac{\sum_{j \in \mathcal{N}_{\text{ref}}} \text{ZNCC}(i, j, d, \vec{n})}{|\mathcal{N}_{\text{ref}}|}$$



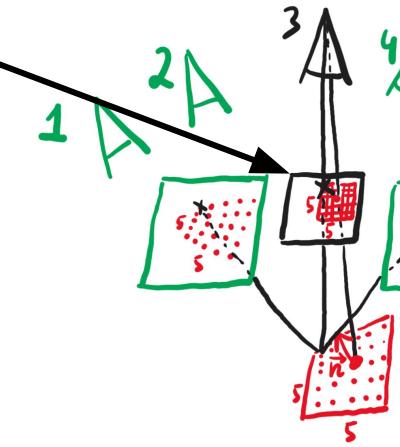
```
1 float zncc(Camera ref, Camera neighbor, int i0, int j0, float d, float3 normal) {  
2       
3       
4       
5       
6       
7       
8       
9       
10      
11      
12      
13      
14      
15      
16    return zncc_res;  
17 }
```

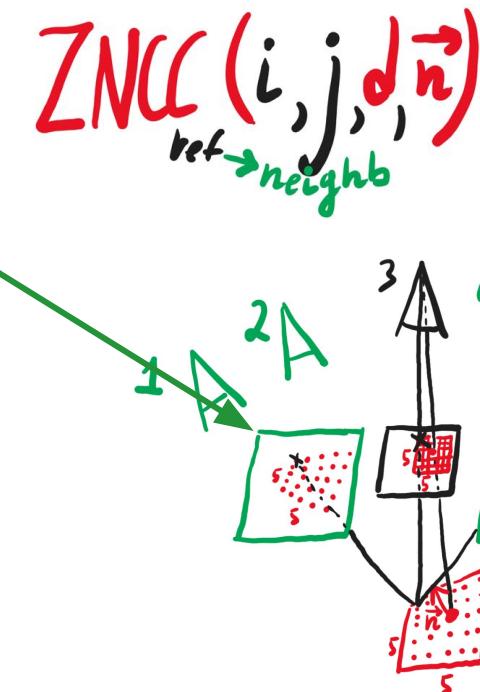
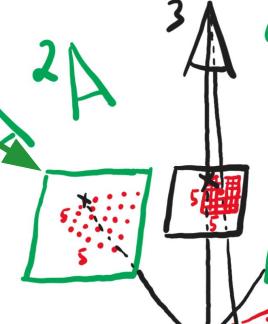
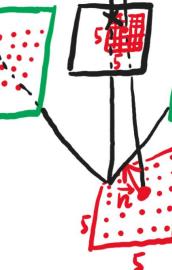
$ZNCC(i, j, d, \vec{n})$
ref \rightarrow neighbor



```
1 float zncc(Camera ref, Camera neighbor, int i0, int j0, float d, float3 normal) {  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16     return zncc_res;  
17 }
```

$ZNCC(i, j, d, \vec{n})$
ref \rightarrow neighbor

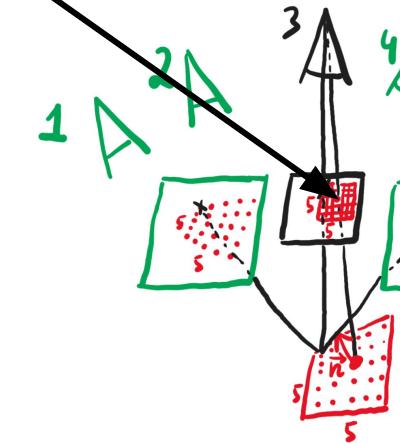


```
1 float zncc(Camera ref, Camera neighbor, int i0, int j0, float d, float3 normal) {  
2       
3           
4       
5       
6       
7       
8       
9       
10      
11      
12      
13      
14      
15      
16     return zncc_res;  
17 }
```

```
1 float zncc(Camera ref, Camera neigh, int i0, int j0, float d, float3 normal) {  
2     // Центральный пиксель патча  
3     // ZNCC(i, j, d, n)  
4     // ref -> neigh  
5     // 1 2 3 4 5  
6     // 1 2 3 4 5  
7     // 1 2 3 4 5  
8     // 1 2 3 4 5  
9     // 1 2 3 4 5  
10    // 1 2 3 4 5  
11    // 1 2 3 4 5  
12    // 1 2 3 4 5  
13    // 1 2 3 4 5  
14    // 1 2 3 4 5  
15    // 1 2 3 4 5  
16    return zncc_res;  
17}
```

центральный пиксель патча

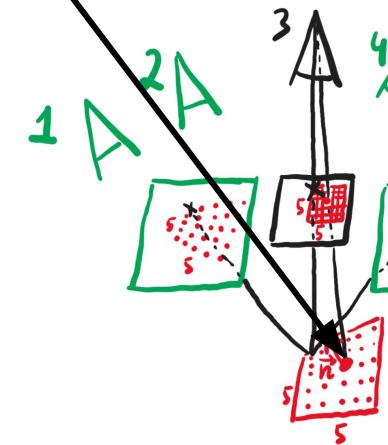
$ZNCC(i, j, d, n)$



```
1 float zncc(Camera ref, Camera neigh, int i0, int j0, float d, float3 normal) {  
2  
3     текущая гипотеза  
4     задает точку на плоскости патча  
5     и нормаль плоскости патча  
6  
7     -  
8  
9     -  
10  
11  
12  
13  
14     -  
15     -  
16     return zncc_res;  
17 }
```

текущая гипотеза
задает точку на плоскости патча
и нормаль плоскости патча

$ZNCC(i, j, d, \vec{n})$



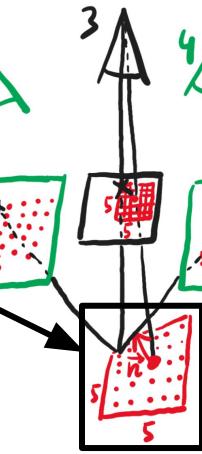
```

1 float zncc(Camera ref, Camera neighbor, int i0, int j0, float d, float3 normal) {
2     float3 XYZ0 = ref.unproject(i0, j0, d);
3     Plane plane(XYZ0, normal);
4     float zncc_res = ...;
5
6     // текущая гипотеза
7     // задает плоскость патча
8
9
10
11
12
13
14
15
16     return zncc_res;
17 }
```

$ZNCC(i, j, d, \vec{n})$

ref \rightarrow neighbor

текущая гипотеза
задает плоскость патча

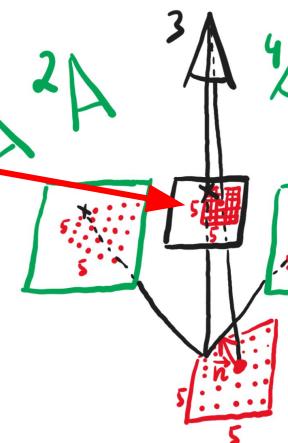


```

1 float zncc(Camera ref, Camera neighbor, int i0, int j0, float d, float3 normal) {
2     float3 XYZ0 = ref.unproject(i0, j0, d);
3     Plane plane(XYZ0, normal);
4     float zncc_res = ...;
5     for (int dj = -2; dj <= 2; ++dj) {
6         for (int di = -2; di <= 2; ++di) {
7             int i1 = i0 + di;
8             int j1 = j0 + dj;
9
10
11
12
13
14     }
15 }
16 return zncc_res;
17 }
```

$ZNCC(i, j, d, \vec{n})$

ref \rightarrow neighbor

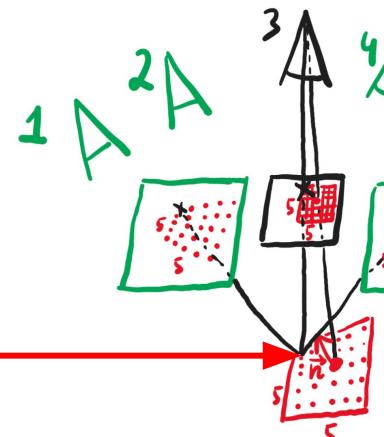


```

1 float zncc(Camera ref, Camera neighbor, int i0, int j0, float d, float3 normal) {
2     float3 XYZ0 = ref.unproject(i0, j0, d);
3     Plane plane(XYZ0, normal);
4     float zncc_res = ...;
5     for (int dj = -2; dj <= 2; ++dj) {
6         for (int di = -2; di <= 2; ++di) {
7             int i1 = i0 + di;
8             int j1 = j0 + dj;
9
10
11
12
13
14     }
15 }
16 return zncc_res;
17 }
```

ZNCC(i, j, d, \vec{n})

ref \rightarrow neighbor

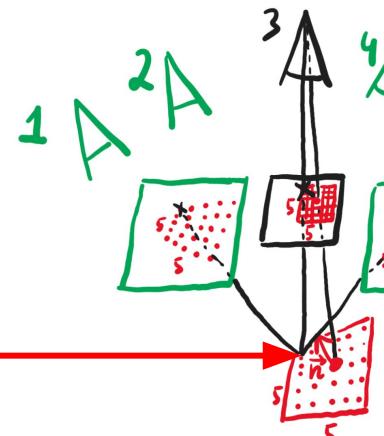


```

1 float zncc(Camera ref, Camera neigh, int i0, int j0, float d, float3 normal) {
2     float3 XYZ0 = ref.unproject(i0, j0, d);
3     Plane plane(XYZ0, normal);
4     float zncc_res = ...;
5     for (int dj = -2; dj <= 2; ++dj) {
6         for (int di = -2; di <= 2; ++di) {
7             int i1 = i0 + di;
8             int j1 = j0 + dj;
9             float3 XYZ1 = ref.unproject(i1, j1, d); ????
10
11            как найти 3D координаты точки?
12
13        }
14    }
15
16    return zncc_res;
17 }
```

ZNCC(i, j, d, \vec{n})

ref \rightarrow neigh



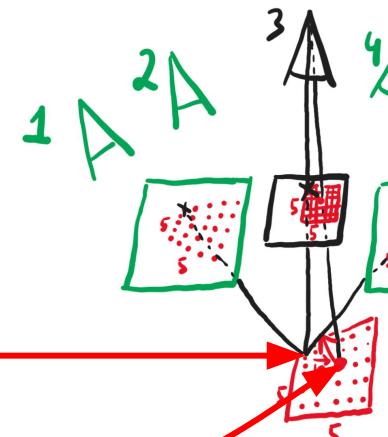
```

1 float zncc(Camera ref, Camera neigh, int i0, int j0, float d, float3 normal) {
2     float3 XYZ0 = ref.unproject(i0, j0, d);
3     Plane plane(XYZ0, normal);
4     float zncc_res = ...;
5     for (int dj = -2; dj <= 2; ++dj) {
6         for (int di = -2; di <= 2; ++di) {
7             int i1 = i0 + di;
8             int j1 = j0 + dj;
9             float3 XYZ1 = ref.unproject(i1, j1, d); ???
10
11
12
13
14     }
15
16     return zncc_res;
17 }
```

$ZNCC(i, j, d, \vec{n})$

ref \rightarrow neigh

как найти 3D координаты точки?



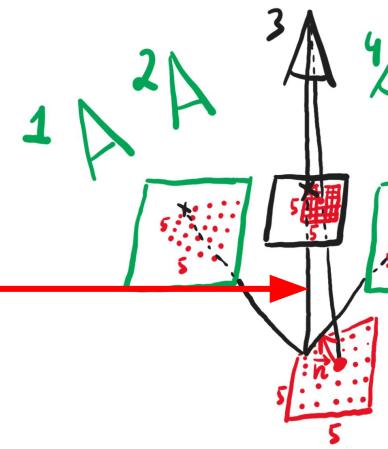
```

1 float zncc(Camera ref, Camera neighbor, int i0, int j0, float d, float3 normal) {
2     float3 XYZ0 = ref.unproject(i0, j0, d);
3     Plane plane(XYZ0, normal);
4     float zncc_res = ...;
5     for (int dj = -2; dj <= 2; ++dj) {
6         for (int di = -2; di <= 2; ++di) {
7             int i1 = i0 + di;
8             int j1 = j0 + dj;
9             float3 XYZ1 = ref.unproject(i1, j1, d);
10            Ray ray(ref.center(), XYZ1 - ref.center());
11
12        }
13    }
14
15    return zncc_res;
16
17 }

```

ZNCC(i, j, d, \vec{n})

ref \rightarrow neighbor

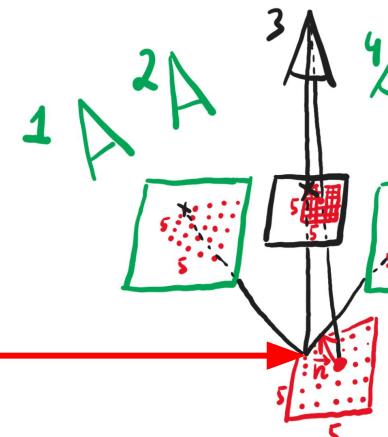


```

1 float zncc(Camera ref, Camera neighbor, int i0, int j0, float d, float3 normal) {
2     float3 XYZ0 = ref.unproject(i0, j0, d);
3     Plane plane(XYZ0, normal);
4     float zncc_res = ...;
5     for (int dj = -2; dj <= 2; ++dj) {
6         for (int di = -2; di <= 2; ++di) {
7             int i1 = i0 + di;
8             int j1 = j0 + dj;
9             float3 XYZ1 = ref.unproject(i1, j1, d);
10            Ray ray(ref.center(), XYZ1 - ref.center());
11            float3 XYZ2 = intersect(ray, plane);
12
13        }
14    }
15
16    return zncc_res;
17 }
```

$ZNCC(i, j, d, \vec{n})$

ref \rightarrow neighbor



```

1 float zncc(Camera ref, Camera neighb, int i0, int j0, float d, float3 normal) {
2     float3 XYZ0 = ref.unproject(i0, j0, d);
3     Plane plane(XYZ0, normal);
4     float zncc_res = ...;
5     for (int dj = -2; dj <= 2; ++dj) {
6         for (int di = -2; di <= 2; ++di) {
7             int i1 = i0 + di;
8             int j1 = j0 + dj;
9             float3 XYZ1 = ref.unproject(i1, j1, d);
10            Ray ray(ref.center(), XYZ1 - ref.center());
11            float3 XYZ2 = intersect(ray, plane);
12            int i2, j2 = neighb.project(XYZ2); // Projected coordinates
13            zncc_res += addPixelToZNCC(ref, i1, j1, neighb, i2, j2);
14        }
15    }
16    return zncc_res;
17 }
```

ZNCC(i, j, d, \vec{n})

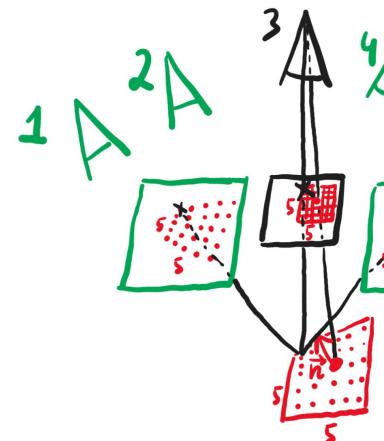
ref \rightarrow neighb

```

1 float zncc(Camera ref, Camera neighb, int i0, int j0, float d, float3 normal) {
2     float3 XYZ0 = ref.unproject(i0, j0, d);
3     Plane plane(XYZ0, normal);
4     float zncc_res = ...;
5     for (int dj = -2; dj <= 2; ++dj) {
6         for (int di = -2; di <= 2; ++di) {
7             int i1 = i0 + di;
8             int j1 = j0 + dj;
9             float3 XYZ1 = ref.unproject(i1, j1, d);
10            Ray ray(ref.center(), XYZ1 - ref.center());
11            float3 XYZ2 = intersect(ray, plane);
12            int i2, j2 = neighb.project(XYZ2);
13            zncc_res += addPixelToZNCC(ref, i1, j1, neighb, i2, j2);
14        }
15    }
16    return zncc_res;
17 }
```

$ZNCC(i, j, d, \vec{n})$

ref \rightarrow neighb

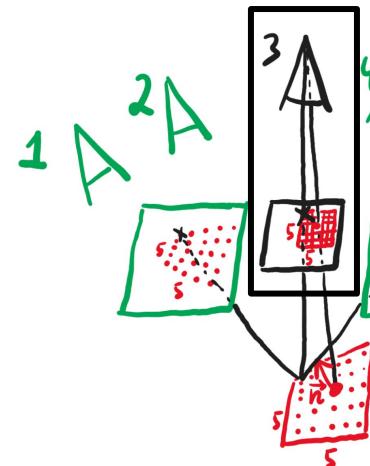


```

1 float zncc(Camera ref, Camera neighb, int i0, int j0, float d, float3 normal) {
2     float3 XYZ0 = ref.unproject(i0, j0, d);
3     Plane plane(XYZ0, normal);
4     float zncc_res = ...;
5     for (int dj = -2; dj <= 2; ++dj) {
6         for (int di = -2; di <= 2; ++di) {
7             int i1 = i0 + di;
8             int j1 = j0 + dj;
9             float3 XYZ1 = ref.unproject(i1, j1, d);
10            Ray ray(ref.center(), XYZ1 - ref.center());
11            float3 XYZ2 = intersect(ray, plane);
12            int i2, j2 = neighb.project(XYZ2);
13            zncc_res += addPixelToZNCC(ref, i1, j1, neighb, i2, j2);
14        }
15    }
16    return zncc_res;
17 }
```

$ZNCC(i, j, d, \vec{n})$

ref \rightarrow neighb

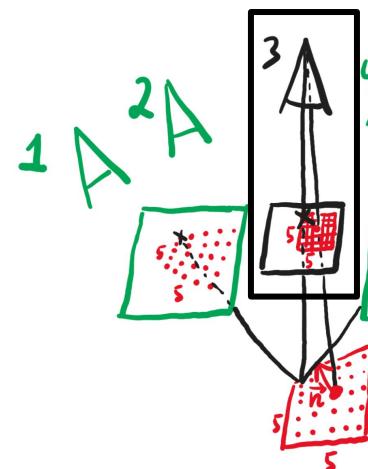


```

1 float zncc(Camera ref, Camera neighb, int i0, int j0, float d, float3 normal) {
2     float3 XYZ0 = ref.unproject(i0, j0, d);
3     Plane plane(XYZ0, normal);
4     float zncc_res = ...;
5     for (int dj = -2; dj <= 2; ++dj) {
6         for (int di = -2; di <= 2; ++di) {
7             int i1 = i0 + di;
8             int j1 = j0 + dj; У всех ли камер есть центр?
9             float3 XYZ1 = ref.unproject(i1, j1, d);
10            Ray ray(ref.center(), XYZ1 - ref.center());
11            float3 XYZ2 = intersect(ray, plane);
12            int i2, j2 = neighb.project(XYZ2);
13            zncc_res += addPixelToZNCC(ref, i1, j1, neighb, i2, j2);
14        }
15    }
16    return zncc_res;
17 }
```

ZNCC(i, j, d, \vec{n})

ref \rightarrow neighb

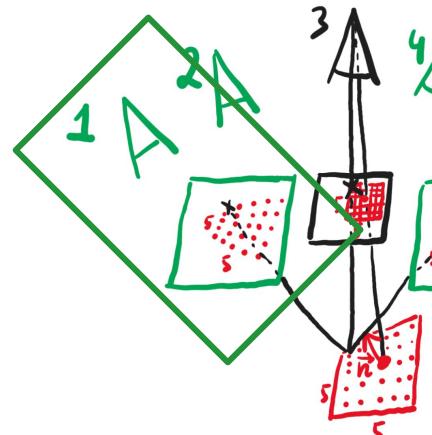


```

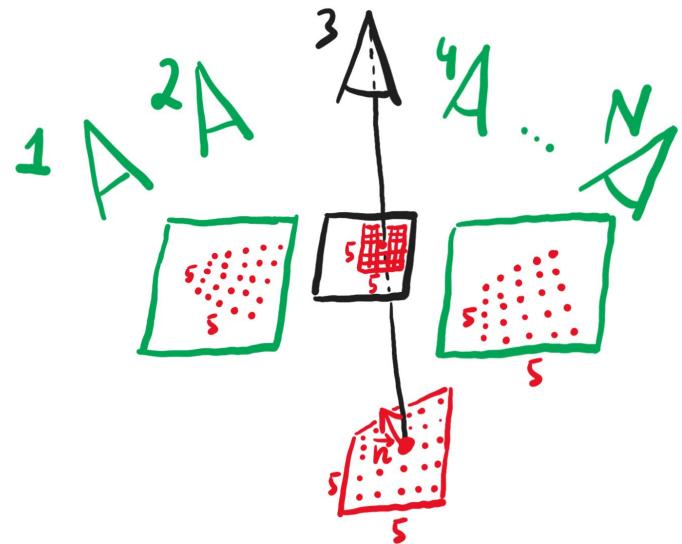
1 float zncc(Camera ref, Camera neighb, int i0, int j0, float d, float3 normal) {
2     float3 XYZ0 = ref.unproject(i0, j0, d);
3     Plane plane(XYZ0, normal);
4     float zncc_res = ...;
5     for (int dj = -2; dj <= 2; ++dj) {
6         for (int di = -2; di <= 2; ++di) {
7             int i1 = i0 + di;
8             int j1 = j0 + dj;
9             float3 XYZ1 = ref.unproject(i1, j1, d);
10            Ray ray(ref.center(), XYZ1 - ref.center());
11            float3 XYZ2 = intersect(ray, plane);
12            int i2, j2 = neighb.project(XYZ2);
13            zncc_res += addPixelToZNCC(ref, i1, j1, neighb, i2, j2);
14        }
15    }
16    return zncc_res;
17 }
```

$ZNCC(i, j, d, \vec{n})$

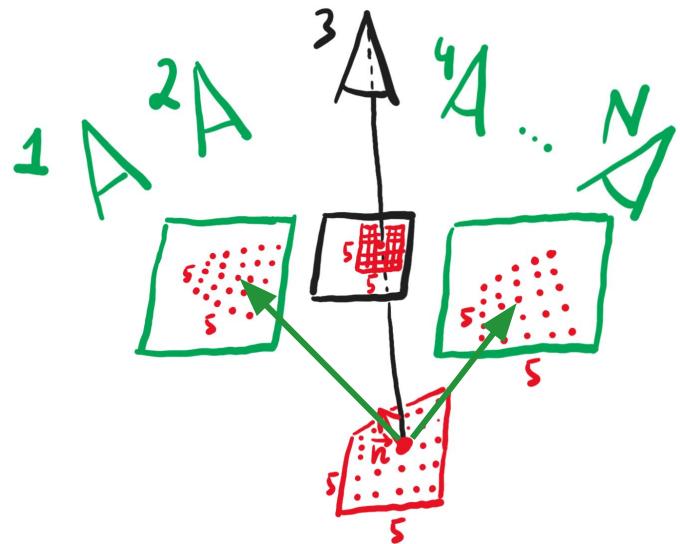
ref \rightarrow neighb



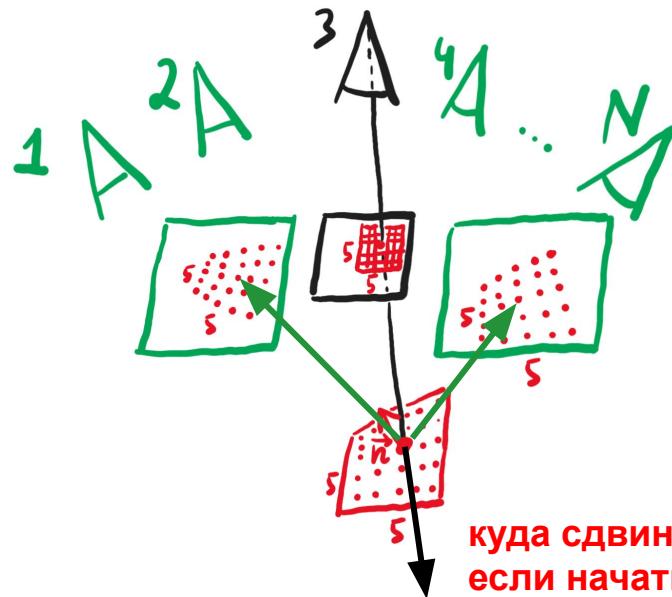
Напоминание



Напоминание

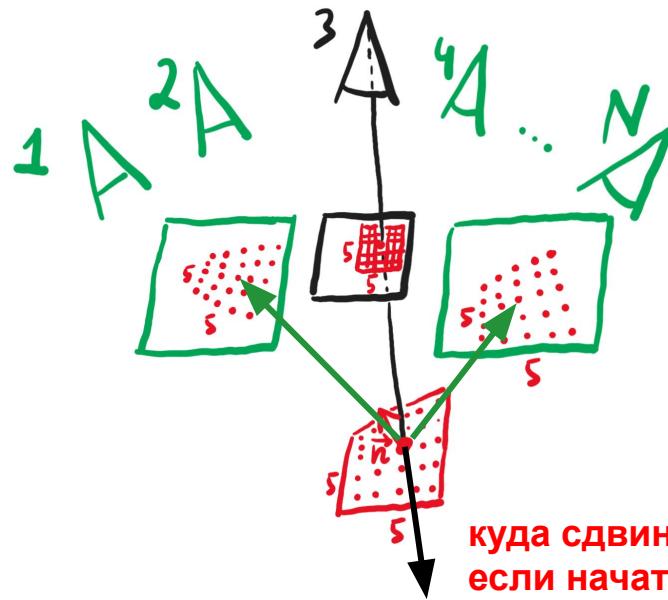


Напоминание



куда сдвинутся проекции центра патча,
если начать увеличивать глубину d ?

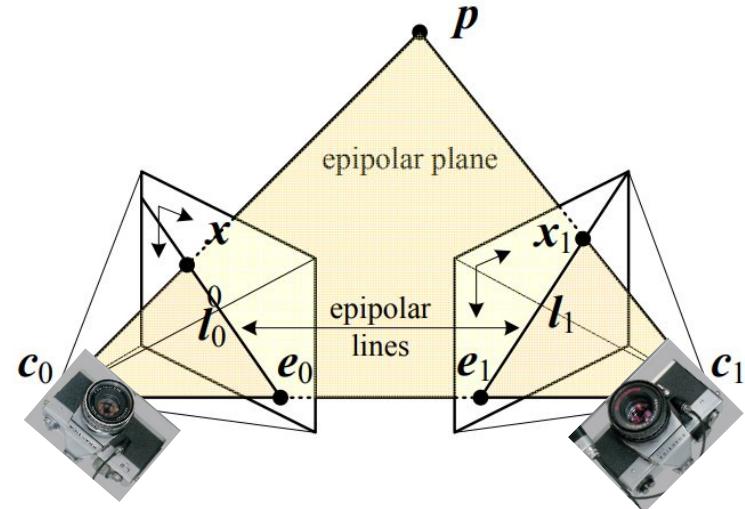
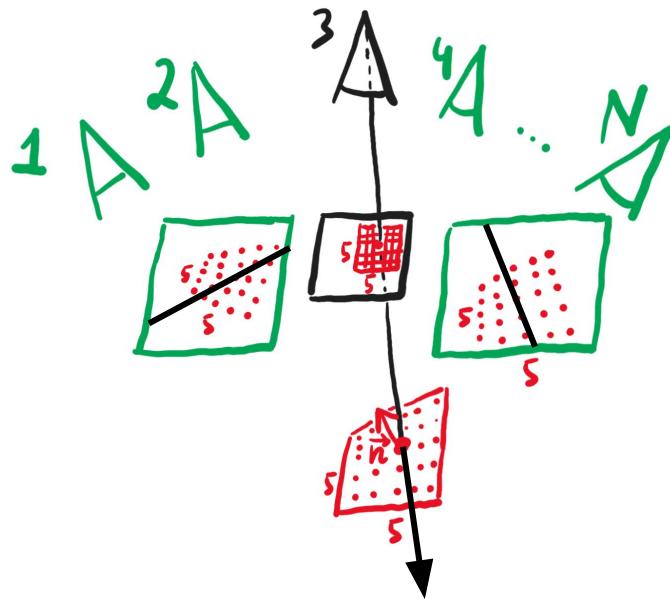
Напоминание

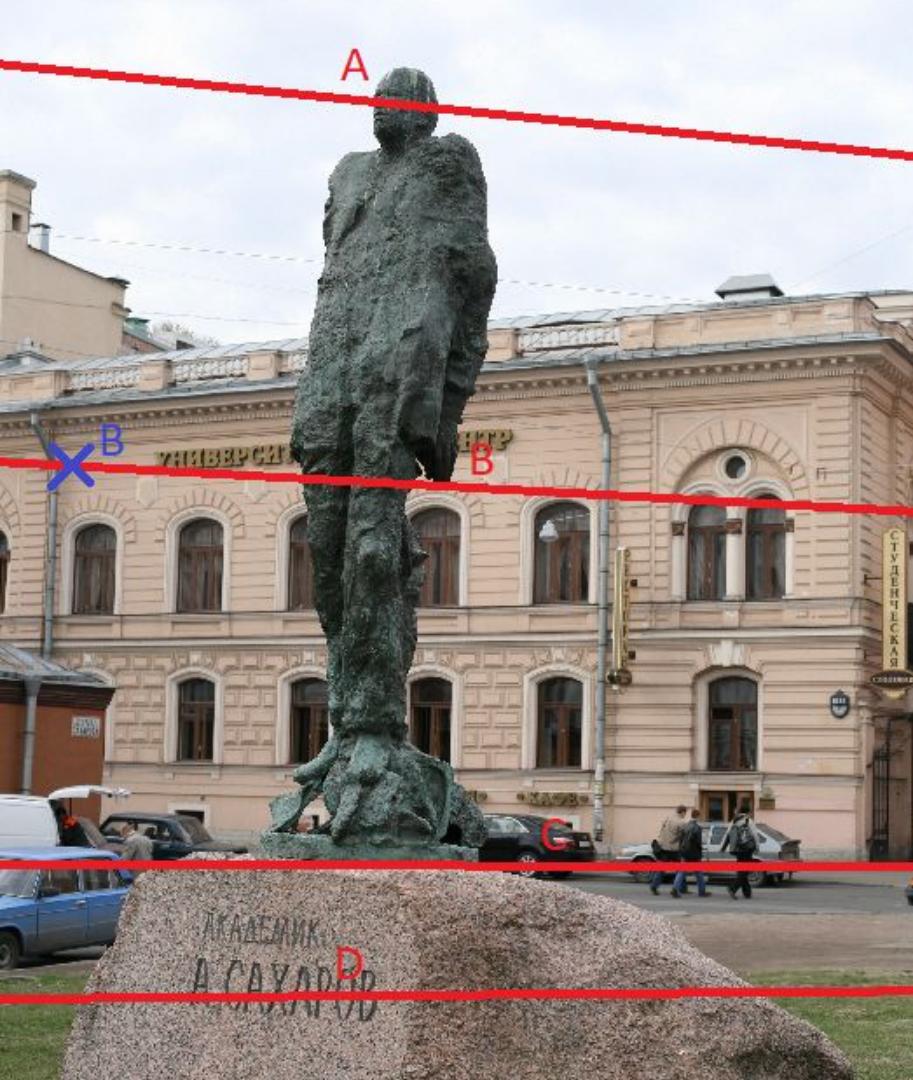


куда сдвинутся проекции центра патча,
если начать увеличивать глубину d ?

является ли множество проекций всех
возможных глубин - прямой?

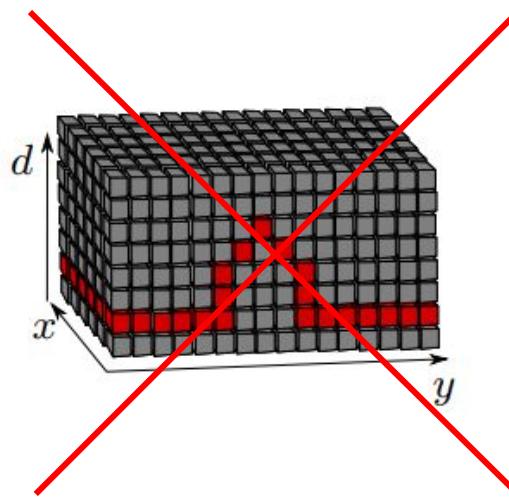
Напоминание про эпиполярную геометрию





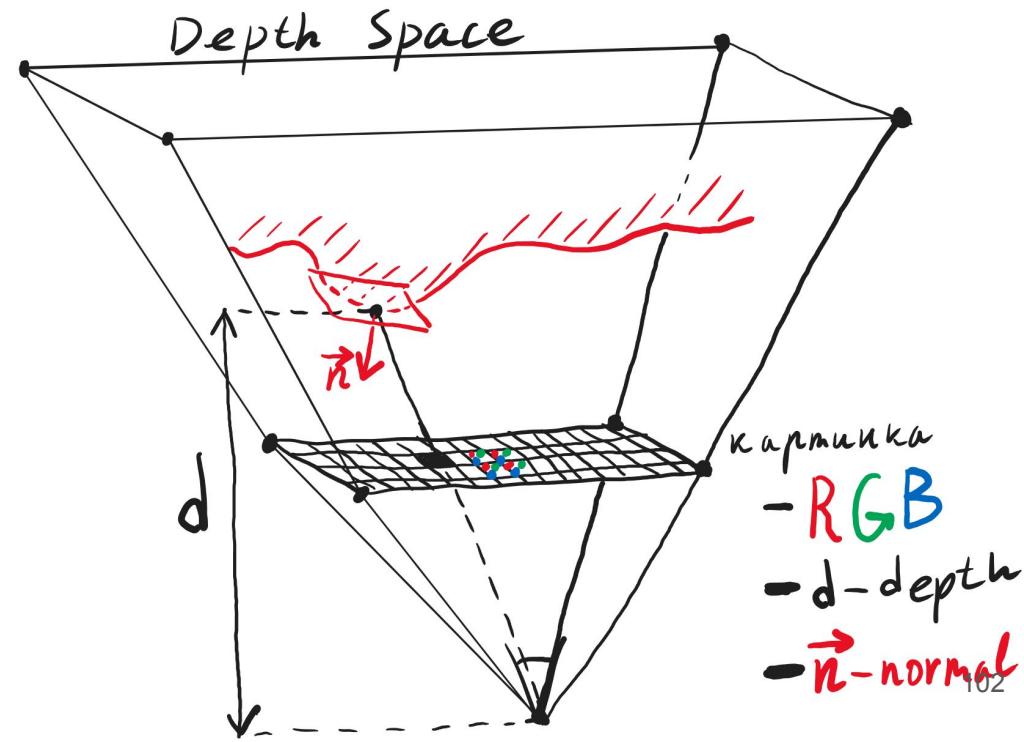
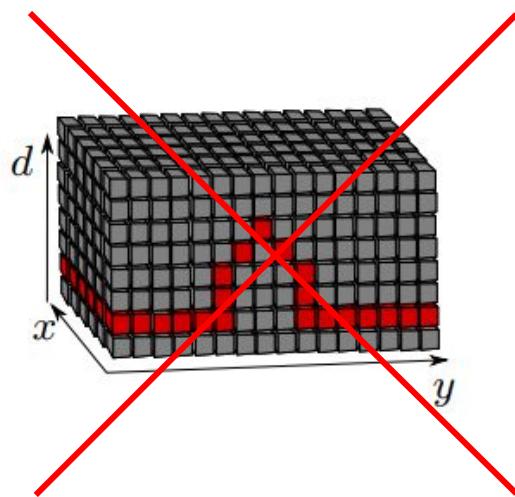
Что мы уже обсудили?

- 1) В гипотезе пикселя: вместо диспаритета - **что?**



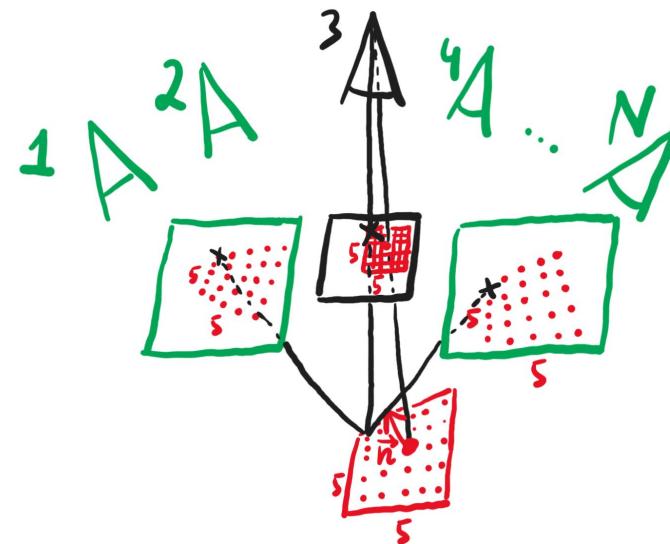
Что мы уже обсудили?

- 1) В гипотезе пикселя: вместо диспаритета - глубина + нормаль



Что мы уже обсудили?

- 1) В гипотезе пикселя: вместо диспаритета - глубина + нормаль
- 2) В оценке гипотезы пикселя участвую **все соседние** фотографии



Что мы уже обсудили?

- 1) В гипотезе пикселя: вместо диспаритета - **глубина + нормаль**
- 2) В оценке гипотезы пикселя участвую **все соседние** фотографии
- 3) Осталось придумать - **откуда брать эти гипотезы?**

Что мы уже обсудили?

- 1) В гипотезе пикселя: вместо диспаритета - **глубина + нормаль**
- 2) В оценке гипотезы пикселя участвую **все соседние** фотографии
- 3) Осталось придумать - **откуда брать эти гипотезы?**
 - полный перебор глубин и нормалей? слишком медленно

Что мы уже обсудили?

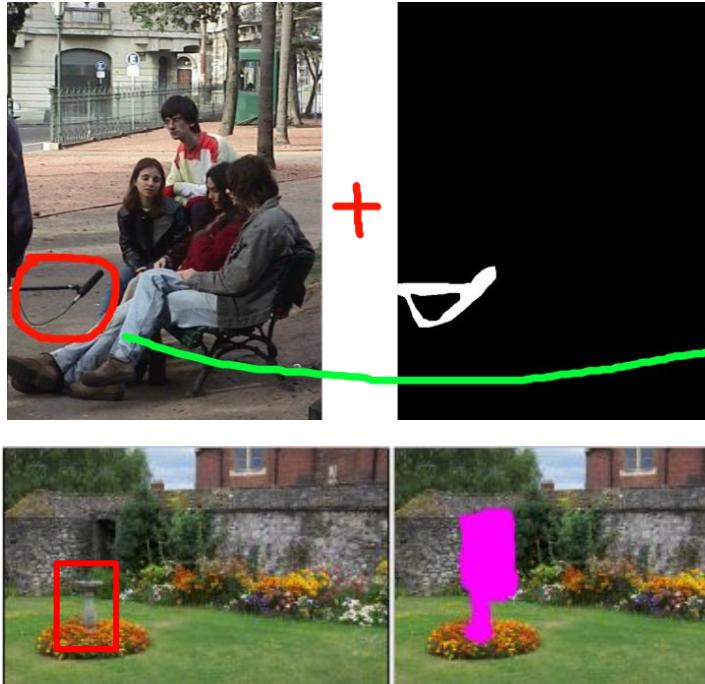
- 1) В гипотезе пикселя: вместо диспаритета - **глубина + нормаль**
- 2) В оценке гипотезы пикселя участвую **все соседние** фотографии
- 3) Осталось придумать - **откуда брать эти гипотезы?**
 - полный перебор глубин и нормалей? слишком медленно
 - случайные гипотезы? каковы шансы что мы найдем правильную?

PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing (*Barnes et. al., 2009*)



(d) input

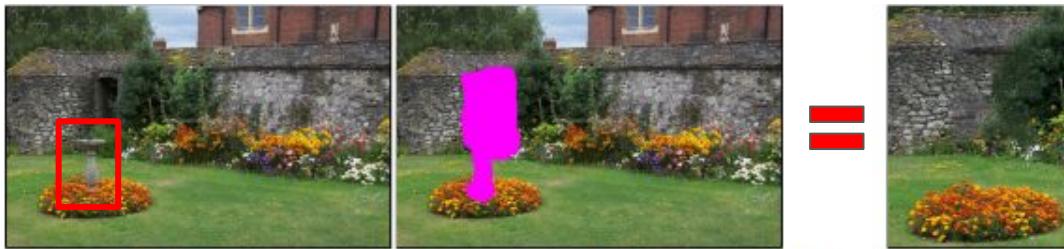
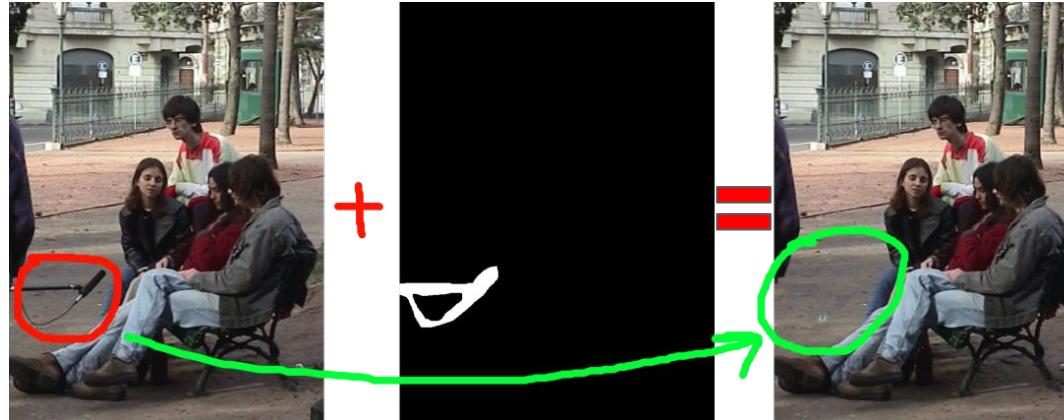
PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing (*Barnes et. al., 2009*)



(d) input

(e) hole

PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing (*Barnes et. al., 2009*)

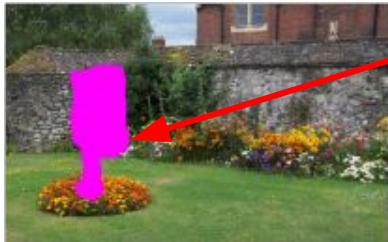


(d) input

(e) hole

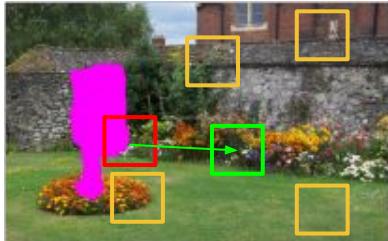
(f) completion (close up)

PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing (*Barnes et. al., 2009*)



Как выбрать какой цвет у пикселя на краю удаленной зоны?

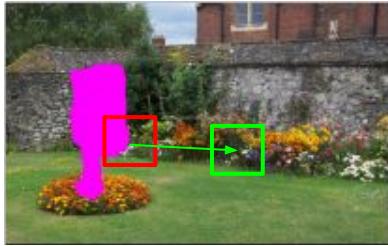
PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing (*Barnes et. al., 2009*)



Найдем в другой части картинки патч, который больше всего похож на нас.

- Как оценить похожесть?

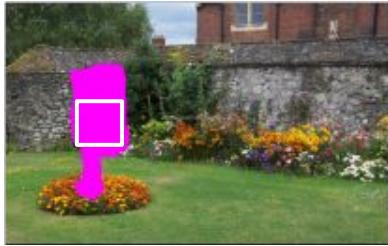
PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing (*Barnes et. al., 2009*)



Найдем в другой части картинки патч, который больше всего похож на нас.

- Как оценить похожесть?
- Все ли пиксели патчей должны вносить вклад в похожесть?

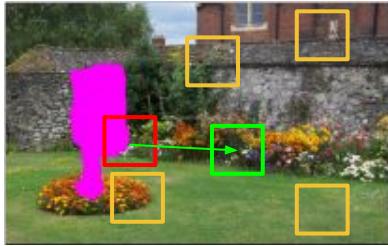
PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing (*Barnes et. al., 2009*)



Найдем в другой части картинки патч, который больше всего похож на нас.

- Как оценить похожесть?
- Все ли пиксели патчей должны вносить вклад в похожесть?
- А что делать с пикселями чей патч целиком удален?

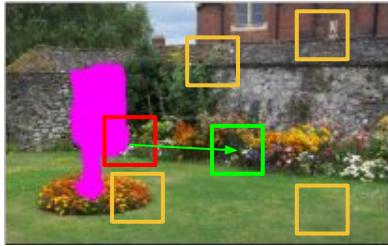
PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing (*Barnes et. al., 2009*)



Найдем в другой части картинки патч, который больше всего похож на нас.

- Как оценить похожесть?
- Все ли пиксели патчей должны вносить вклад в похожесть?
- А что делать с пикселями чей патч целиком удален?
- **Как выглядит алгоритм в целом?**

PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing (*Barnes et. al., 2009*)

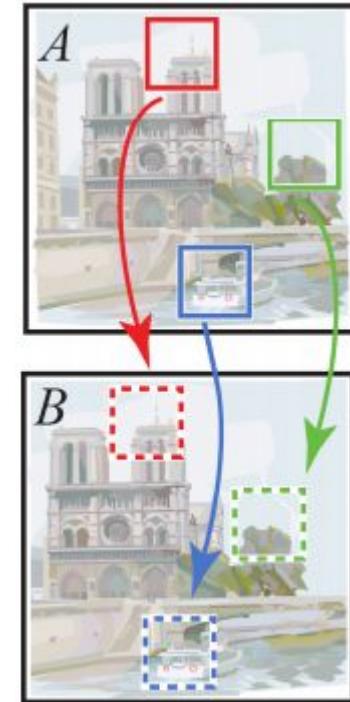


Найдем в другой части картинки патч, который больше всего похож на нас.

- Как оценить похожесть?
- Все ли пиксели патчей должны вносить вклад в похожесть?
- А что делать с пикселями чей патч целиком удален?
- Как выглядит алгоритм в целом?
- **Быстро ли он работает?**

PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing (*Barnes et. al., 2009*)

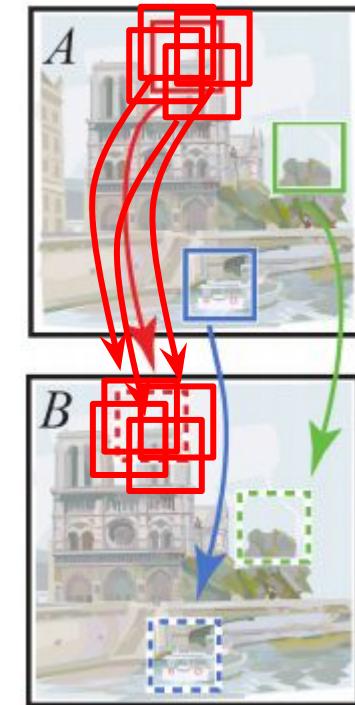
Хотим для каждого патча в А найти самый похожий патч в В.



PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing (*Barnes et. al., 2009*)

Хотим для каждого патча в А найти самый похожий патч в В.

Natural structure of images: правильный ответ обычно содержит большие связные регионы сопоставления. Можем увеличить эффективность выполняя поиск зависимо от соседей. Т.е. вдохновляясь их успехами.

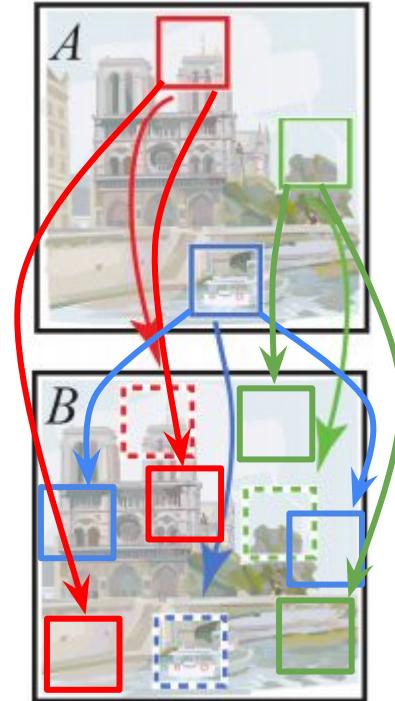


PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing (*Barnes et. al., 2009*)

Хотим для каждого патча в А найти самый похожий патч в В.

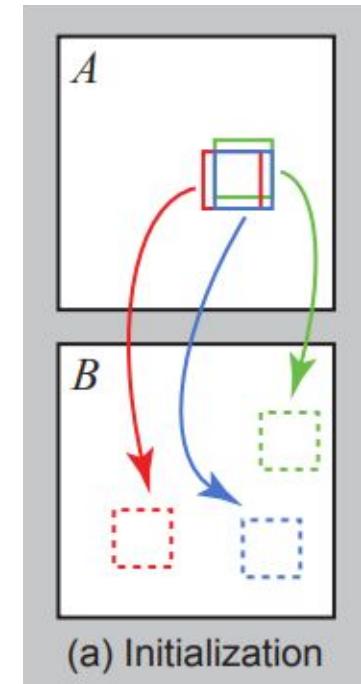
Natural structure of images: правильный ответ обычно содержит большие связные регионы сопоставления. Можем увеличить эффективность выполняя поиск зависимо от соседей. Т.е. вдохновляясь их успехами.

The law of large numbers: одна случайная гипотеза не угадает никогда. Но если каждый пиксель посмотрит в случайное место - кто-то да угадает! Дополнительные итерации увеличивают шансы еще больше.



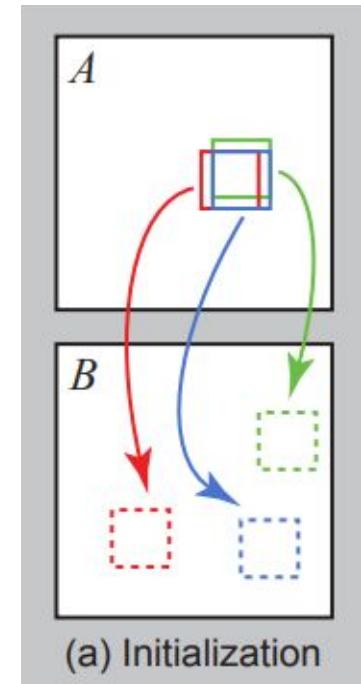
PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing (*Barnes et. al., 2009*)

- 1) В каждом пикселе случайная гипотеза (dx, dy)



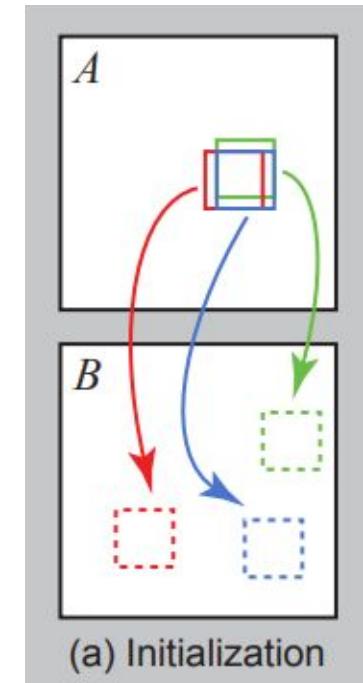
PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing (*Barnes et. al., 2009*)

- 1) В каждом пикселе случайная гипотеза (dx, dy)
- 2) Несколько итераций делаем:



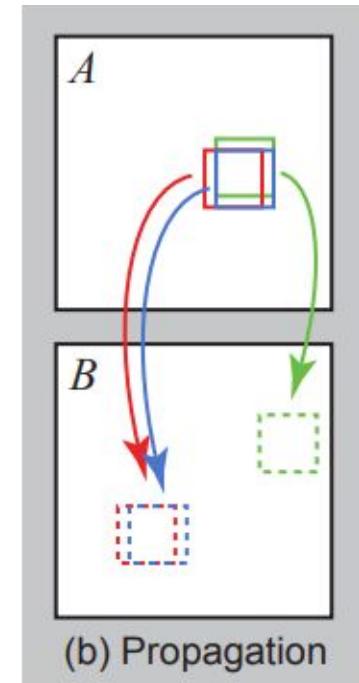
PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing (Barnes et. al., 2009)

- 1) В каждом пикселе случайная гипотеза (dx, dy)
- 2) Несколько итераций делаем:
 - 2.1) **Propagation:** примеряем **на себя** гипотезы **соседей**.
Т.е. используем их как источник хорошей идеи.



PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing (Barnes et. al., 2009)

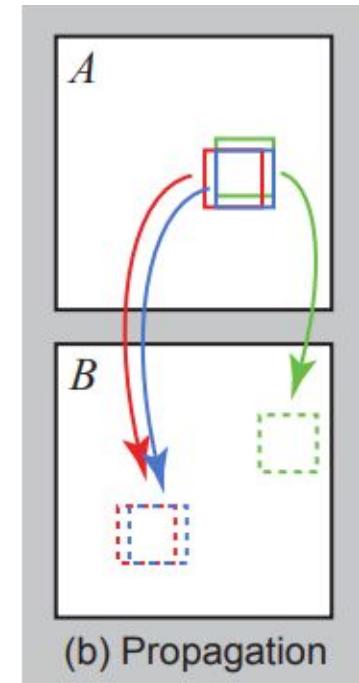
- 1) В каждом пикселе случайная гипотеза (dx , dy)
- 2) Несколько итераций делаем:
 - 2.1) **Propagation:** примеряем на себя гипотезы соседей.
Т.е. используем их как источник хорошей идеи.



PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing (Barnes et. al., 2009)

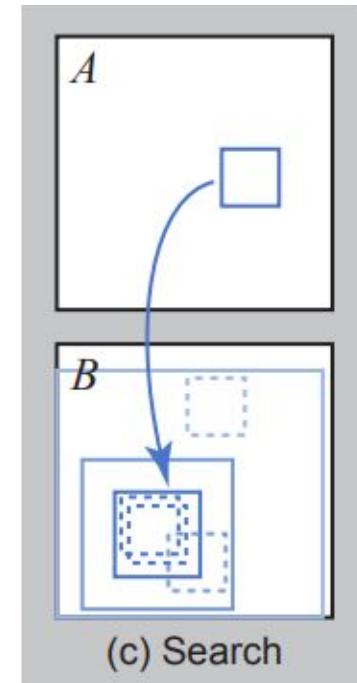
- 1) В каждом пикселе случайная гипотеза (dx , dy)
- 2) Несколько итераций делаем:
 - 2.1) **Propagation:** примеряем на себя гипотезы соседей.
Т.е. используем их как источник хорошей идеи.

А почему синяя стрелка показывает не туда же, куда и красная?



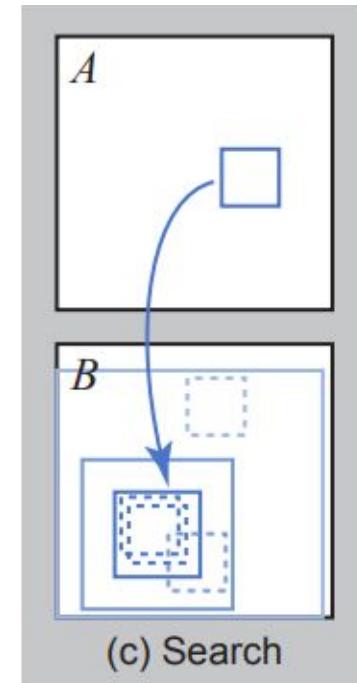
PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing (Barnes et. al., 2009)

- 1) В каждом пикселе случайная гипотеза (dx, dy)
- 2) Несколько итераций делаем:
 - 2.1) **Propagation:** примеряем на себя гипотезы соседей.
Т.е. используем их как источник хорошей идеи.
 - 2.2) **Refinement:** пробуем улучшить/уточнить свою гипотезу случайными сдвигами:
 - распределение равномерное по всей картинке?



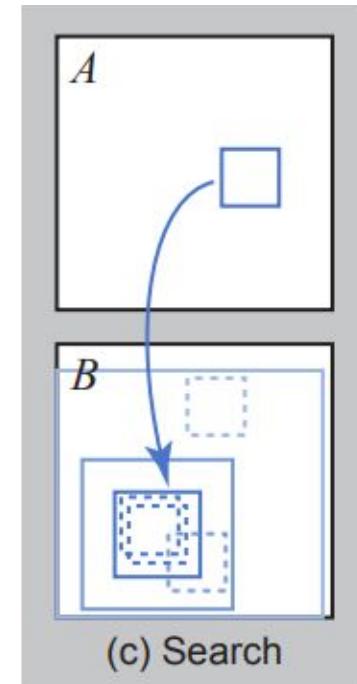
PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing (Barnes et. al., 2009)

- 1) В каждом пикселе случайная гипотеза (dx, dy)
- 2) Несколько итераций делаем:
 - 2.1) **Propagation:** примеряем на себя гипотезы соседей.
Т.е. используем их как источник хорошей идеи.
 - 2.2) **Refinement:** пробуем улучшить/уточнить свою гипотезу случайными сдвигами:
 - малый сдвиг - с большими шансами (**уточнение**)



PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing (Barnes et. al., 2009)

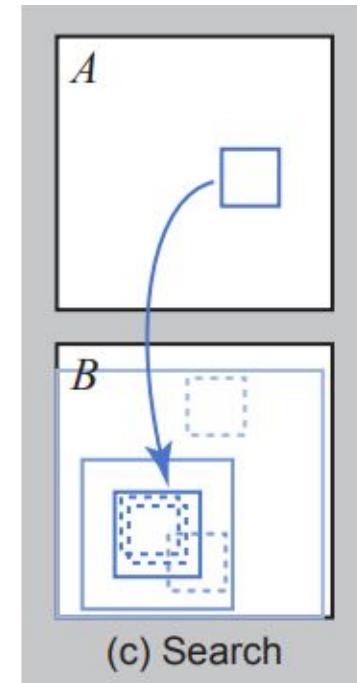
- 1) В каждом пикселе случайная гипотеза (dx , dy)
- 2) Несколько итераций делаем:
 - 2.1) **Propagation:** примеряем на себя гипотезы соседей.
Т.е. используем их как источник хорошей идеи.
 - 2.2) **Refinement:** пробуем улучшить/уточнить свою гипотезу случайными сдвигами:
 - малый сдвиг - с большими шансами (**уточнение**)
 - большой сдвиг - редко (**новые** хорошие идеи)



PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing (Barnes et. al., 2009)

Как ускорить сходимость?

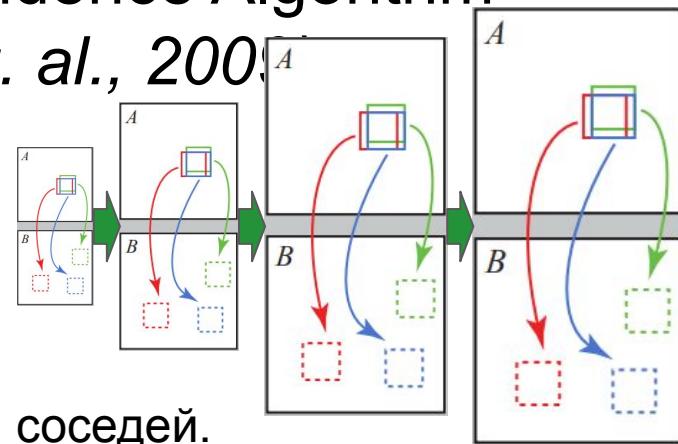
- 1) В каждом пикселе случайная гипотеза (dx , dy)
- 2) Несколько итераций делаем:
 - 2.1) **Propagation:** примеряем на себя гипотезы соседей.
Т.е. используем их как источник хорошей идеи.
 - 2.2) **Refinement:** пробуем улучшить/уточнить свою гипотезу случайными сдвигами:
 - малый сдвиг - с большими шансами (**уточнение**)
 - большой сдвиг - редко (**новые** хорошие идеи)



PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing (Barnes et. al., 2009)

for **level** = 0 ... N (**Coarse-to-Fine**)

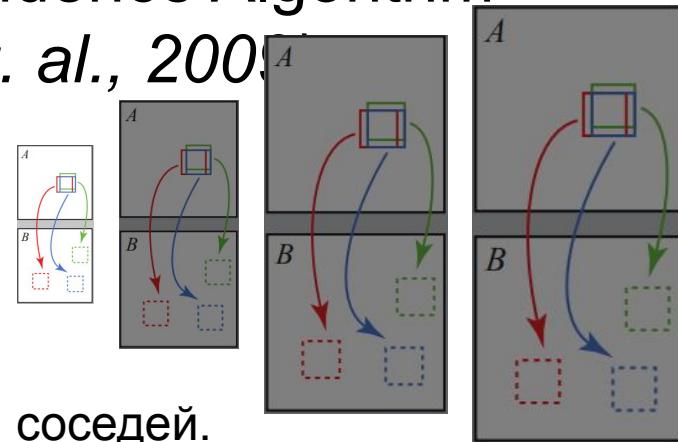
- 1) В каждом пикселе случайная гипотеза (dx , dy)
- 2) Несколько итераций делаем:
 - 2.1) **Propagation:** примеряем на себя гипотезы соседей.
Т.е. используем их как источник хорошей идеи.
 - 2.2) **Refinement:** пробуем улучшить/уточнить свою гипотезу случайными сдвигами:
 - малый сдвиг - с большими шансами (**уточнение**)
 - большой сдвиг - редко (**новые** хорошие идеи)



PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing (Barnes et. al., 2009)

for **level** = 0 ... N (**Coarse-to-Fine**)

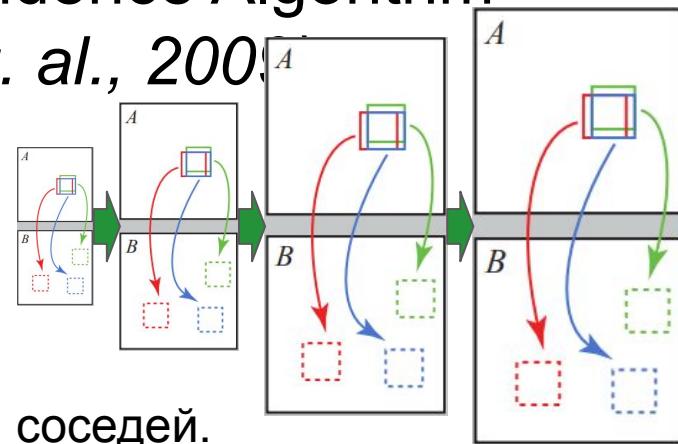
- 1) В каждом пикселе **случайная гипотеза (dx, dy)**
- 2) Несколько итераций делаем:
 - 2.1) **Propagation:** примеряем на себя гипотезы соседей.
Т.е. используем их как источник хорошей идеи.
 - 2.2) **Refinement:** пробуем улучшить/уточнить свою гипотезу случайными сдвигами:
 - малый сдвиг - с большими шансами (**уточнение**)
 - большой сдвиг - редко (**новые** хорошие идеи)



PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing (Barnes et. al., 2009)

for **level** = 0 ... N (**Coarse-to-Fine**)

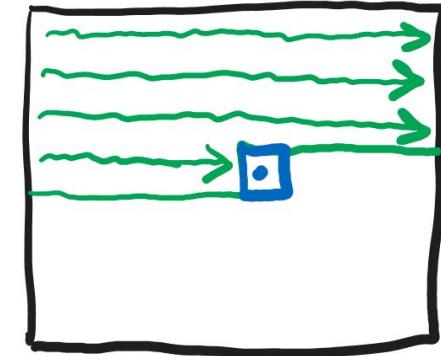
- 1) В каждом пикселе **upscale** / случайное (dx, dy)
- 2) Несколько итераций делаем:
 - 2.1) **Propagation:** примеряем на себя гипотезы соседей.
Т.е. используем их как источник хорошей идеи.
 - 2.2) **Refinement:** пробуем улучшить/уточнить свою гипотезу случайными сдвигами:
 - малый сдвиг - с большими шансами (**уточнение**)
 - большой сдвиг - редко (**новые** хорошие идеи)



PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing (Barnes et. al., 2009)

for **level** = 0 ... N (**Coarse-to-Fine**)

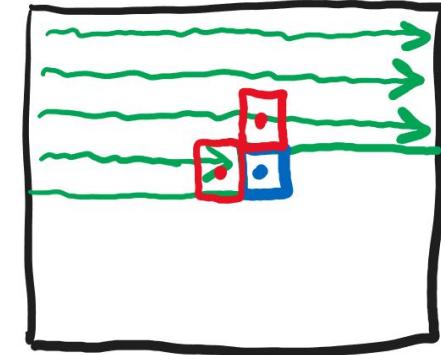
- 1) В каждом пикселе **upscale** / случайное (dx, dy)
- 2) Несколько итераций делаем:
 - 2.1) **Propagation:** примеряем на себя гипотезы соседей.
Т.е. используем их как источник хороший идеи.
При прямом проходе - **???**
 - 2.2) **Refinement:** пробуем улучшить/уточнить свою гипотезу случайными сдвигами:
 - малый сдвиг - с большими шансами (**уточнение**)
 - большой сдвиг - редко (**новые** хорошие идеи)



PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing (Barnes et. al., 2009)

for **level** = 0 ... N (**Coarse-to-Fine**)

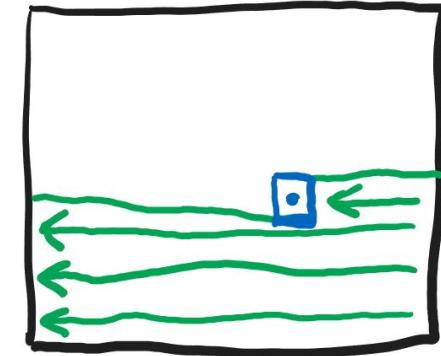
- 1) В каждом пикселе **upscale** / случайное (dx, dy)
- 2) Несколько итераций делаем:
 - 2.1) **Propagation:** примеряем на себя гипотезы соседей.
Т.е. используем их как источник хорошей идеи.
При прямом проходе - **соседи слева/сверху.**
 - 2.2) **Refinement:** пробуем улучшить/уточнить свою гипотезу случайными сдвигами:
 - малый сдвиг - с большими шансами (**уточнение**)
 - большой сдвиг - редко (**новые** хорошие идеи)



PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing (Barnes et. al., 2009)

for **level** = 0 ... N (**Coarse-to-Fine**)

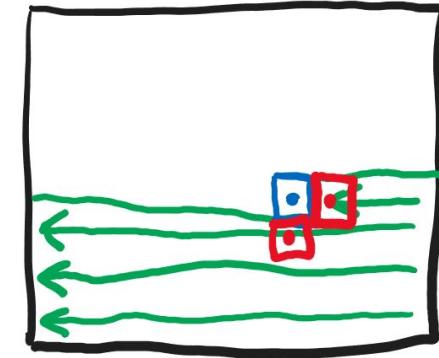
- 1) В каждом пикселе **upscale** / случайное (dx, dy)
- 2) Несколько итераций делаем:
 - 2.1) **Propagation:** примеряем на себя гипотезы соседей.
Т.е. используем их как источник хорошей идеи.
При прямом проходе - соседи слева/сверху.
При обратном проходе - ???
 - 2.2) **Refinement:** пробуем улучшить/уточнить свою гипотезу случайными сдвигами:
 - малый сдвиг - с большими шансами (**уточнение**)
 - большой сдвиг - редко (**новые** хорошие идеи)



PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing (Barnes et. al., 2009)

for **level** = 0 ... N (**Coarse-to-Fine**)

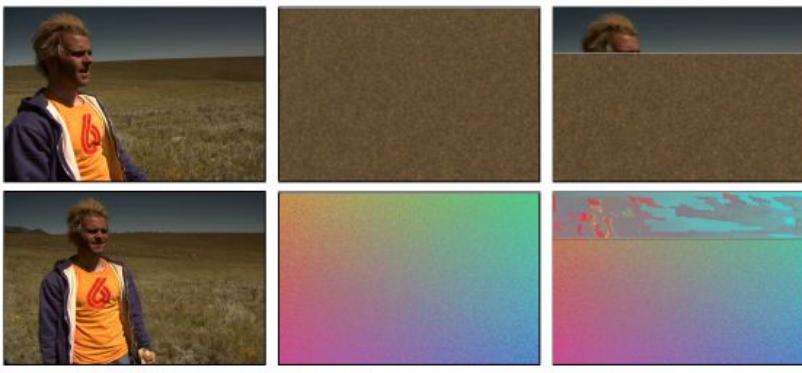
- 1) В каждом пикселе **upscale** / случайное (dx, dy)
- 2) Несколько итераций делаем:
 - 2.1) **Propagation:** примеряем на себя гипотезы соседей.
Т.е. используем их как источник хорошей идеи.
При прямом проходе - соседи слева/сверху.
При обратном проходе - **соседи справа/снизу.**
 - 2.2) **Refinement:** пробуем улучшить/уточнить свою гипотезу случайными сдвигами:
 - малый сдвиг - с большими шансами (**уточнение**)
 - большой сдвиг - редко (**новые** хорошие идеи)





(a) originals

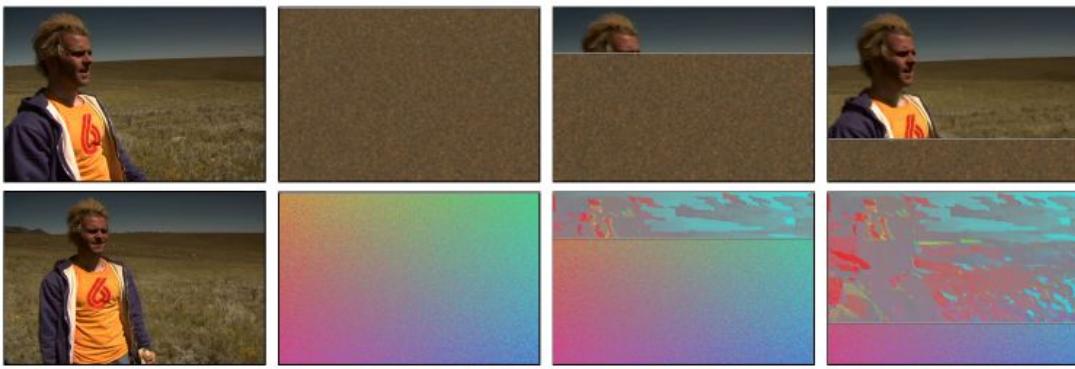
(b) random



(a) originals

(b) random

(c) $\frac{1}{4}$ iteration

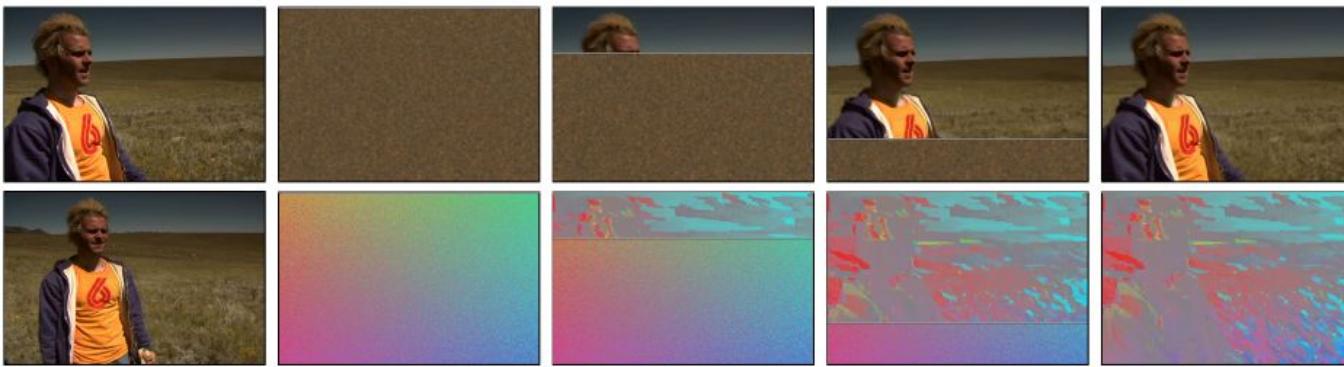


(a) originals

(b) random

(c) $\frac{1}{4}$ iteration

(d) $\frac{3}{4}$ iteration



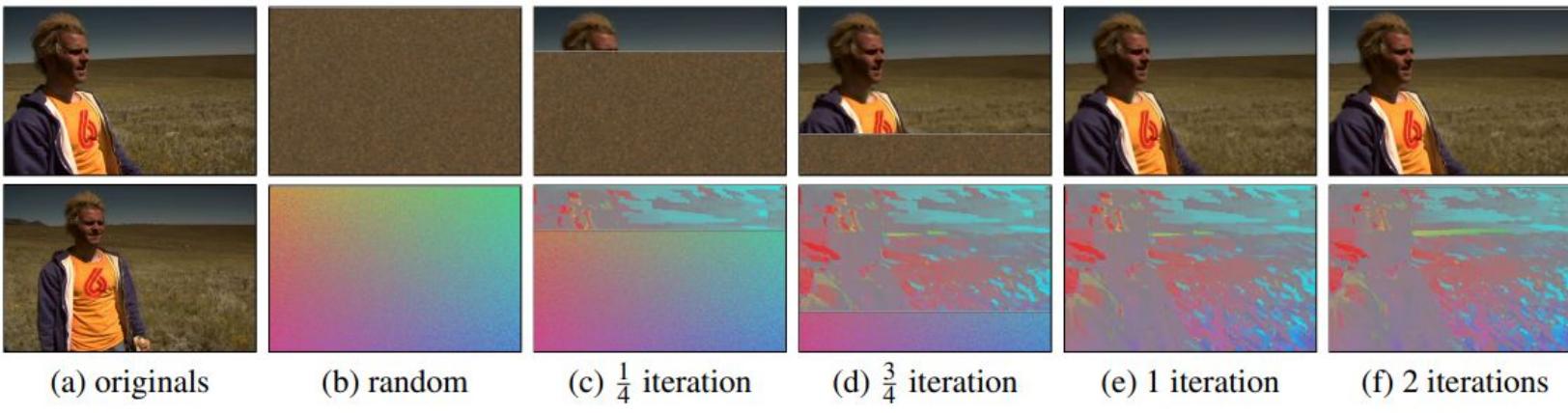
(a) originals

(b) random

(c) $\frac{1}{4}$ iteration

(d) $\frac{3}{4}$ iteration

(e) 1 iteration



(a) originals

(b) random

(c) $\frac{1}{4}$ iteration

(d) $\frac{3}{4}$ iteration

(e) 1 iteration

(f) 2 iterations

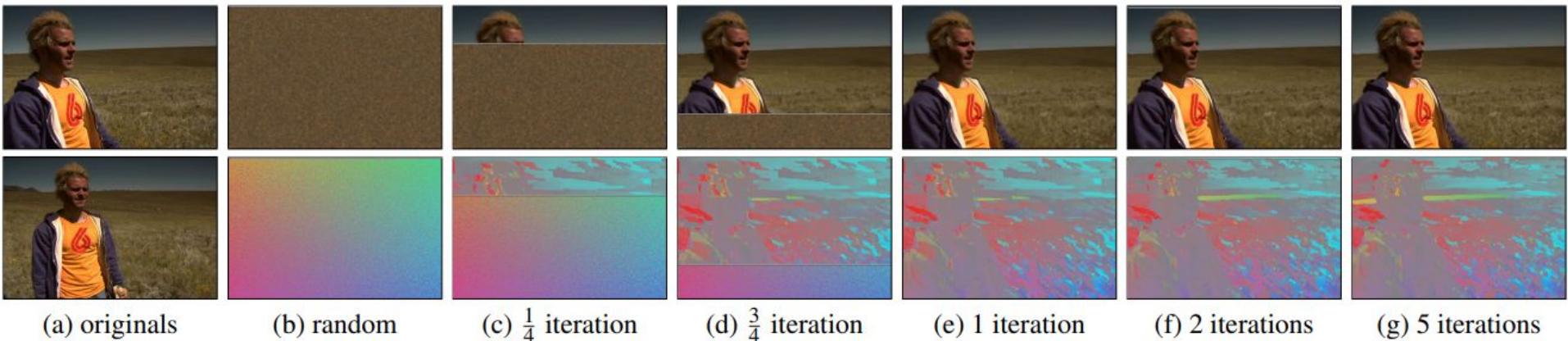
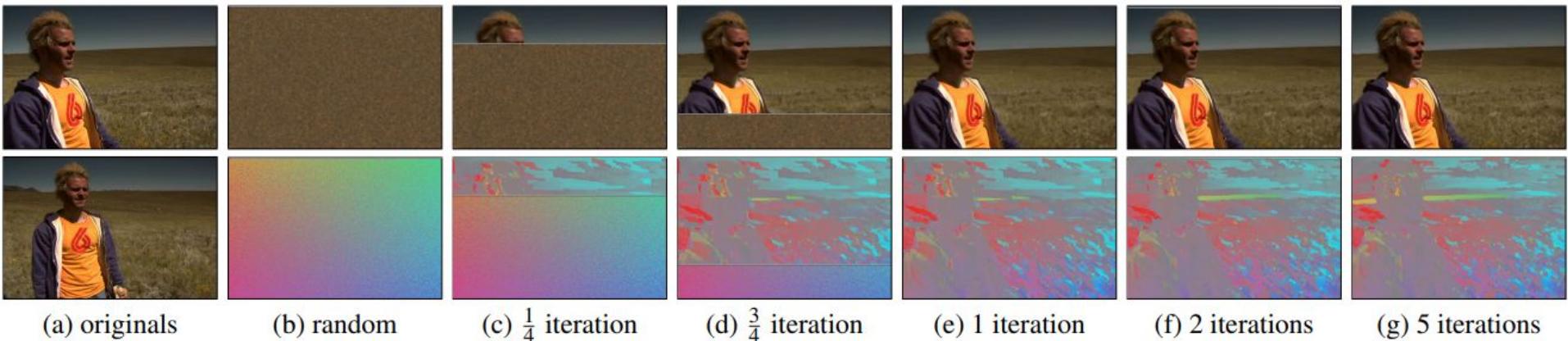


Figure 3: Illustration of convergence. (a) The top image is reconstructed using only patches from the bottom image. (b) above: the reconstruction by the patch “voting” described in Section 4, below: a random initial offset field, with magnitude visualized as saturation and angle visualized as hue. (c) 1/4 of the way through the first iteration, high-quality offsets have been propagated in the region above the current scan line (denoted with the horizontal bar). (d) 3/4 of the way through the first iteration. (e) First iteration complete. (f) Two iterations. (g) After 5 iterations, almost all patches have stopped changing. The tiny orange flowers only find good correspondences in the later iterations.



(a) originals

(b) random

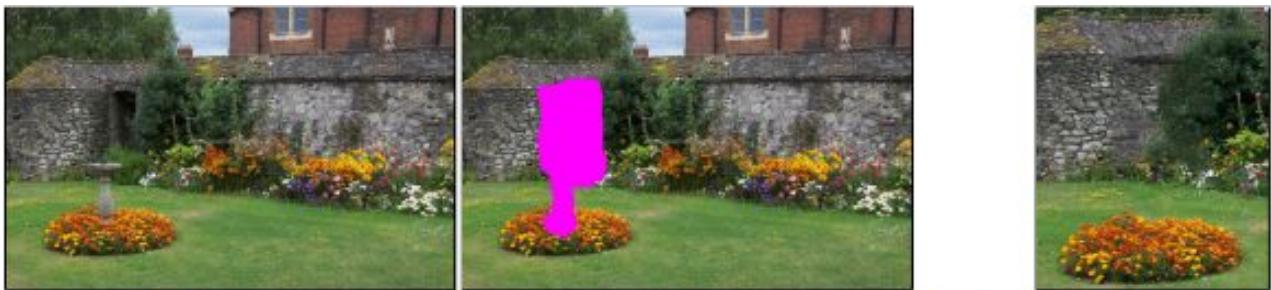
(c) $\frac{1}{4}$ iteration(d) $\frac{3}{4}$ iteration

(e) 1 iteration

(f) 2 iterations

(g) 5 iterations

Figure 3: Illustration of convergence. (a) The top image is reconstructed using only patches from the bottom image. (b) above: the reconstruction by the patch “voting” described in Section 4, below: a random initial offset field, with magnitude visualized as saturation and angle visualized as hue. (c) 1/4 of the way through the first iteration, high-quality offsets have been propagated in the region above the current scan line (denoted with the horizontal bar). (d) 3/4 of the way through the first iteration. (e) First iteration complete. (f) Two iterations. (g) After 5 iterations, almost all patches have stopped changing. The tiny orange flowers only find good correspondences in the later iterations.



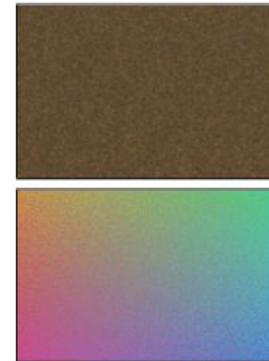
(d) input

(e) hole

(f) completion (close up)

Построение карты глубины методом Patch Match

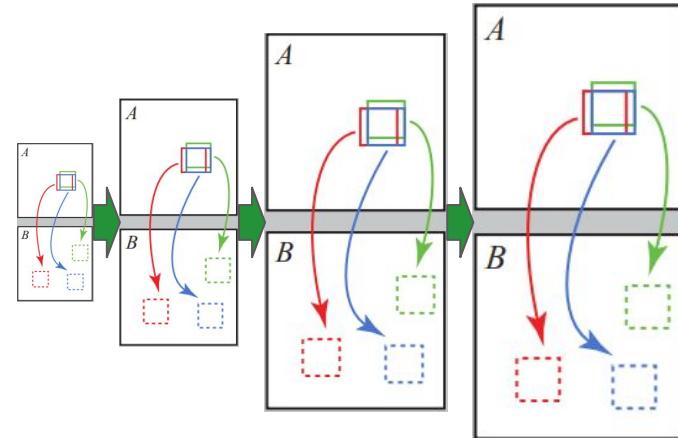
```
depth_map, normal_map = random()
```



Построение карты глубины методом Patch Match

depth_map, normal_map = random()

for level = 0 ... N: (Coarse to Fine)

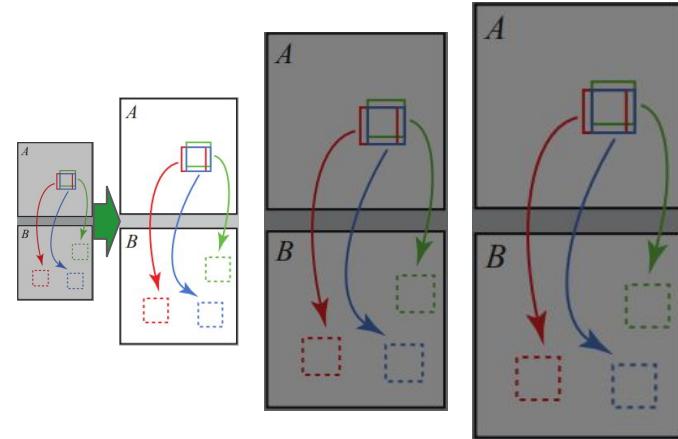


Построение карты глубины методом Patch Match

```
depth_map, normal_map = random()
```

```
for level = 0 ... N:
```

```
    upscale(depth_map, normal_map)
```



Построение карты глубины методом Patch Match

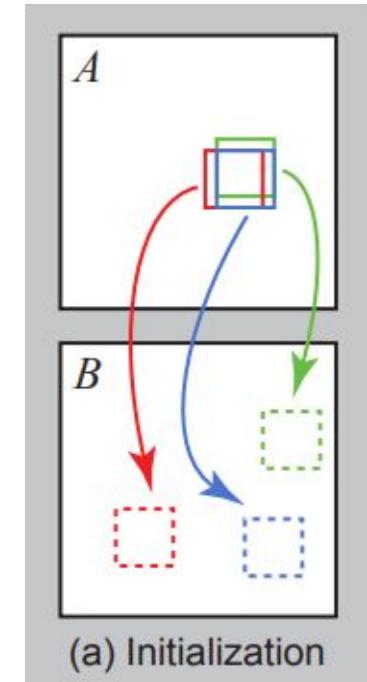
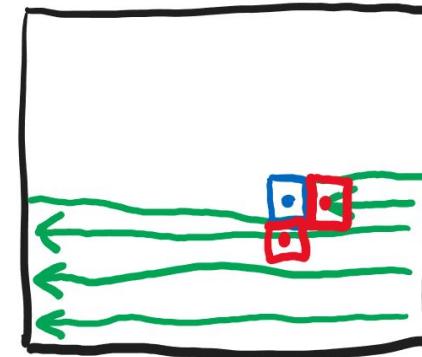
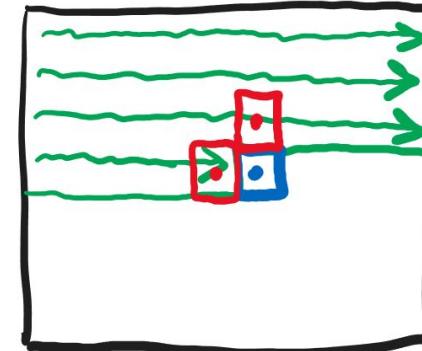
```
depth_map, normal_map = random()
```

```
for level = 0 ... N:
```

```
    upscale(depth_map, normal_map)
```

```
    for iteration = 0 ... 100:
```

```
        propagation()
```



(a) Initialization

Построение карты глубины методом Patch Match

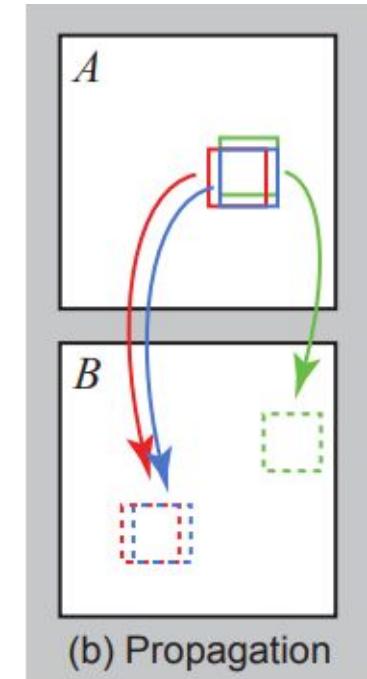
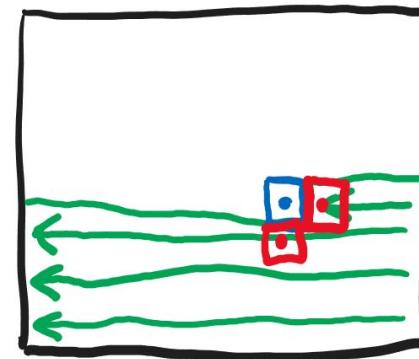
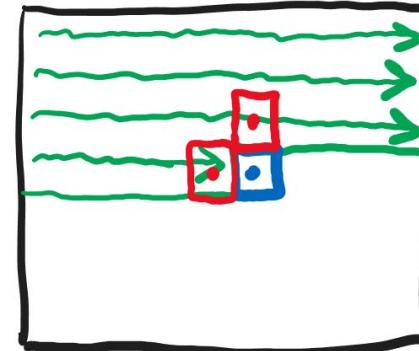
```
depth_map, normal_map = random()
```

```
for level = 0 ... N:
```

```
    upscale(depth_map, normal_map)
```

```
    for iteration = 0 ... 100:
```

```
        propagation()
```



(b) Propagation

Построение карты глубины методом Patch Match

```
depth_map, normal_map = random()
```

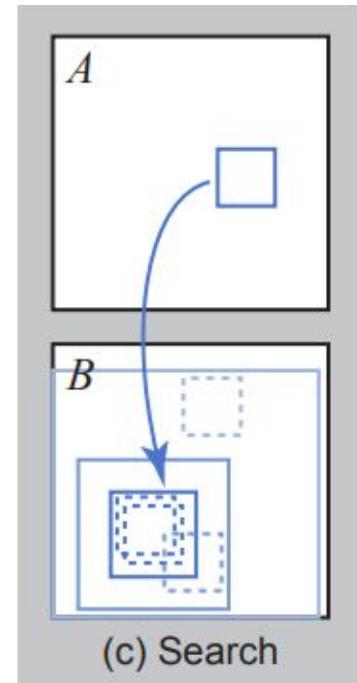
```
for level = 0 ... N:
```

```
    upscale(depth_map, normal_map)
```

```
    for iteration = 0 ... 100:
```

```
        propagation()
```

```
        refinement()
```



Построение карты глубины методом Patch Match

```
depth_map, normal_map = random()
```

```
for level = 0 ... N:
```

```
    upscale(depth_map, normal_map)
```

```
    for iteration = 0 ... 100:
```

```
        propagation()
```

```
        refinement()
```

```
return depth_map, normal_map
```

Построение карты глубины методом Patch Match

Базовые кирпичики:

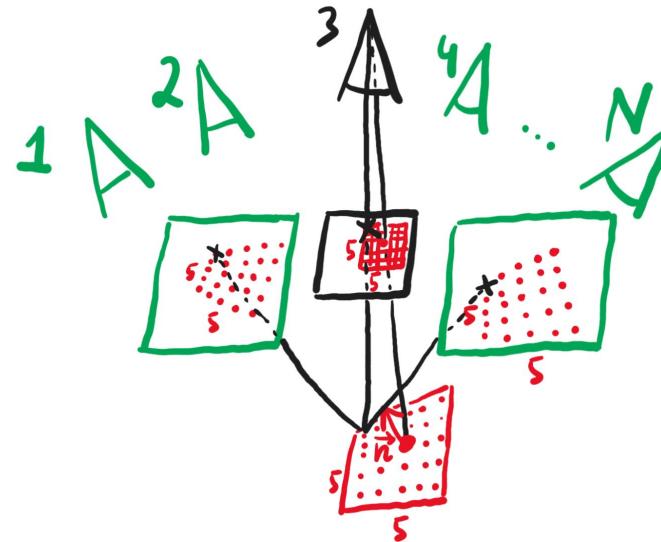
1) Когда мы примеряем на себя **depth + normal** гипотезу...

Построение карты глубины методом Patch Match

Базовые кирпичики:

1) Когда мы примеряем на себя **depth + normal** гипотезу - оцениваем **cost** по всем камерам.

$$cost_{ref}(i, j, \vec{d}, \vec{n}) = \frac{\sum_{j \in N_{ref}} ZNCC(i, j, \vec{d}, \vec{n})}{|N_{ref}|}$$

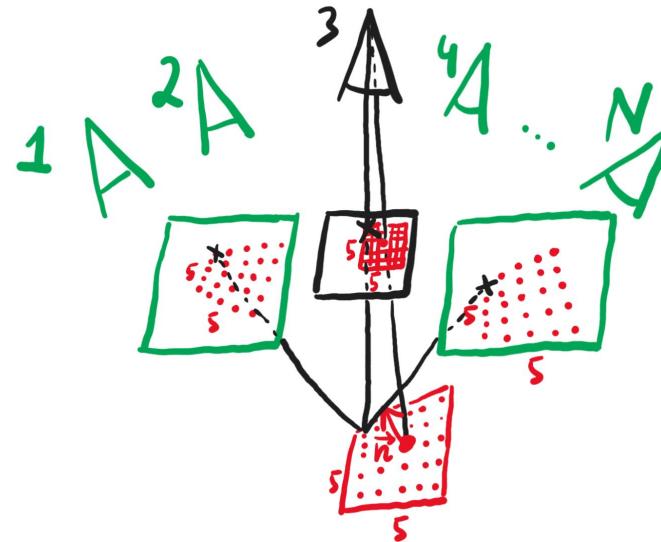


Построение карты глубины методом Patch Match

Базовые кирпичики:

1) Когда мы примеряем на себя **depth + normal** гипотезу - оцениваем **cost** по всем камерам. Если эта оценка **лучше** чем то что до сих пор у нас было - заменяем нашу гипотезу на то что только что примерили.

$$cost_{ref}(i, j, \vec{d}, \vec{n}) = \frac{\sum_{j \in N_{ref}} ZNCC(i, j, \vec{d}, \vec{n})}{|N_{ref}|}$$



Построение карты глубины методом Patch Match

Базовые кирпичики:

- 1) Когда мы примеряем на себя **depth + normal** гипотезу - оцениваем **cost** по всем камерам. Если эта оценка **лучше** чем то что до сих пор у нас было - заменяем нашу гипотезу на то что только что примерили.
- 2) Оценка **cost** сводится к нескольким базовым функциям:
 - $XYZ = \text{camera.unproject}(i, j, d)$

Построение карты глубины методом Patch Match

Базовые кирпичики:

1) Когда мы примеряем на себя **depth + normal** гипотезу - оцениваем **cost** по всем камерам. Если эта оценка **лучше** чем то что до сих пор у нас было - заменяем нашу гипотезу на то что только что примерили.

2) Оценка **cost** сводится к нескольким базовым функциям:

- $XYZ = \text{camera.unproject}(i, j, d)$
- $i, j = \text{camera.project}(XYZ)$

Построение карты глубины методом Patch Match

Базовые кирпичики:

- 1) Когда мы примеряем на себя **depth + normal** гипотезу - оцениваем **cost** по всем камерам. Если эта оценка **лучше** чем то что до сих пор у нас было - заменяем нашу гипотезу на то что только что примерили.
- 2) Оценка **cost** сводится к нескольким базовым функциям:
 - $XYZ = \text{camera.unproject}(i, j, d)$
 - $i, j = \text{camera.project}(XYZ)$
 - $XYZ = \text{intersectPlane}(\text{ray}, \text{Plane}(XYZ, \text{normal}))$

Построение карты глубины методом Patch Match

Базовые кирпичики:

- 1) Когда мы примеряем на себя **depth + normal** гипотезу - оцениваем **cost** по всем камерам. Если эта оценка **лучше** чем то что до сих пор у нас было - заменяем нашу гипотезу на то что только что примерили.
- 2) Оценка **cost** сводится к нескольким базовым функциям:
 - $XYZ = \text{camera.unproject}(i, j, d)$
 - $i, j = \text{camera.project}(XYZ)$
 - $XYZ = \text{intersectPlane}(\text{ray}, \text{Plane}(XYZ, \text{normal}))$

Вопросы?



Полярный Николай
polarnick239@gmail.com