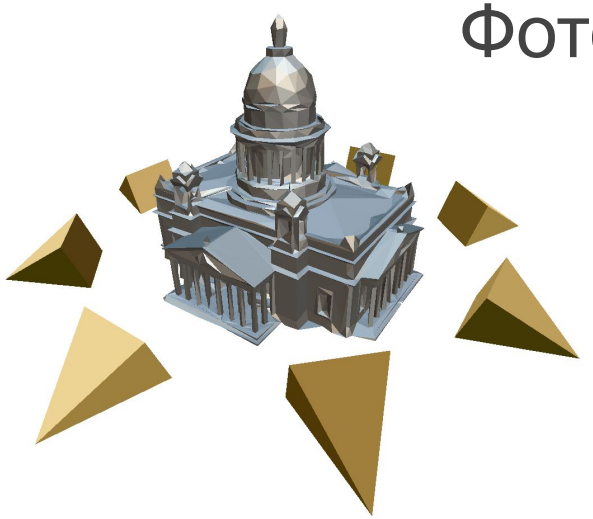
Построение карт глубины. PatchMatch

Фотограмметрия. Лекция 6



- **Gipuma**
- **Colmap**
- **ACMH/ACMM**

Полярный Николай
polarnick239@gmail.com

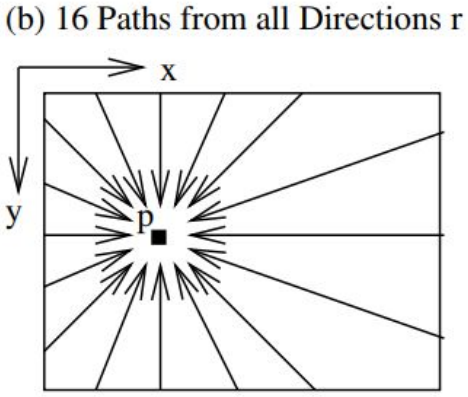
Где мы сейчас?

- Есть разреженное облако 3D положений ключевых точек
- Есть хорошо оптимизированные внутренние калибровки камер (intrinsics)
- Есть хорошо оптимизированные положения и ракурсы камер (extrinsics)

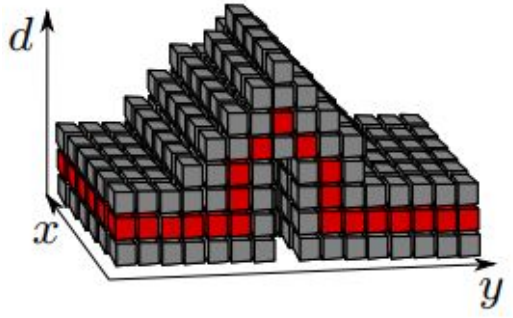
Что хотим дальше?

Очень детальную и точную геометрию поверхности (в виде карт глубины).

Semi-Global Matching (SGM, tSGM)



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



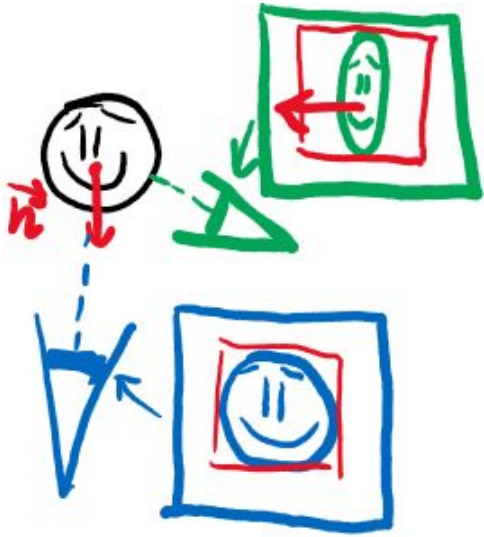
За счет **coarse-to-fine** схемы (tSGM: пирамиды картинок + прогрессивный поиск карты глубины на новой ступени детализации в ограниченном диапазоне диспаритетов) - еще быстрее и, главное, с **гарантиями по памяти**.

Semi-Global Matching (SGM, tSGM)

1) Что делать с тонкими структурами?
(усы котика, перила у лестницы, антены)

2) Как улучшить подсчет **Cost** у патчей наблюдаемых под разным углом? (а не только фронто-параллельные)

3) Что делать с бликами и регулярными паттернами?



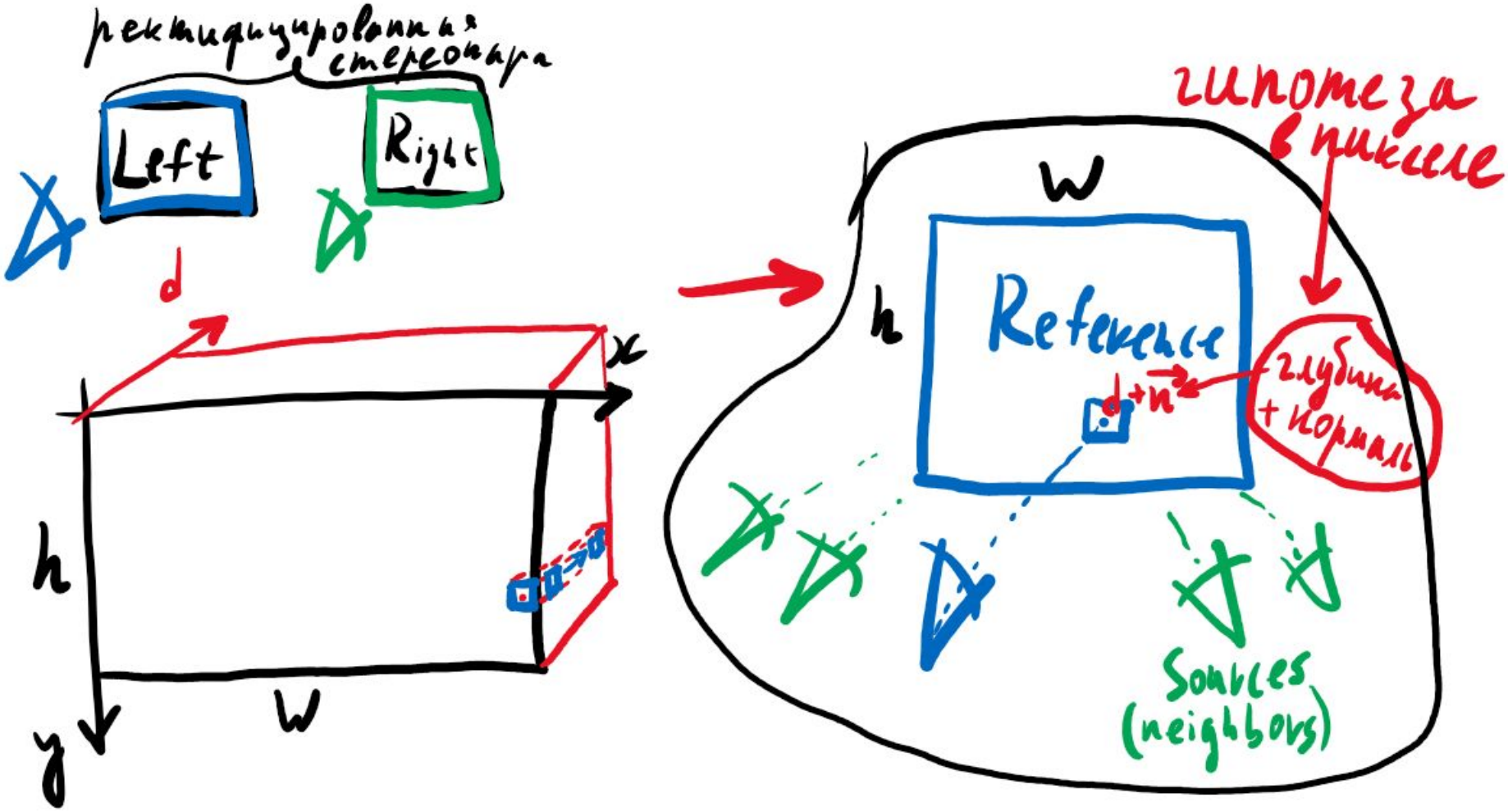
Вывод

1) Чем меньше размер патча - тем лучше!

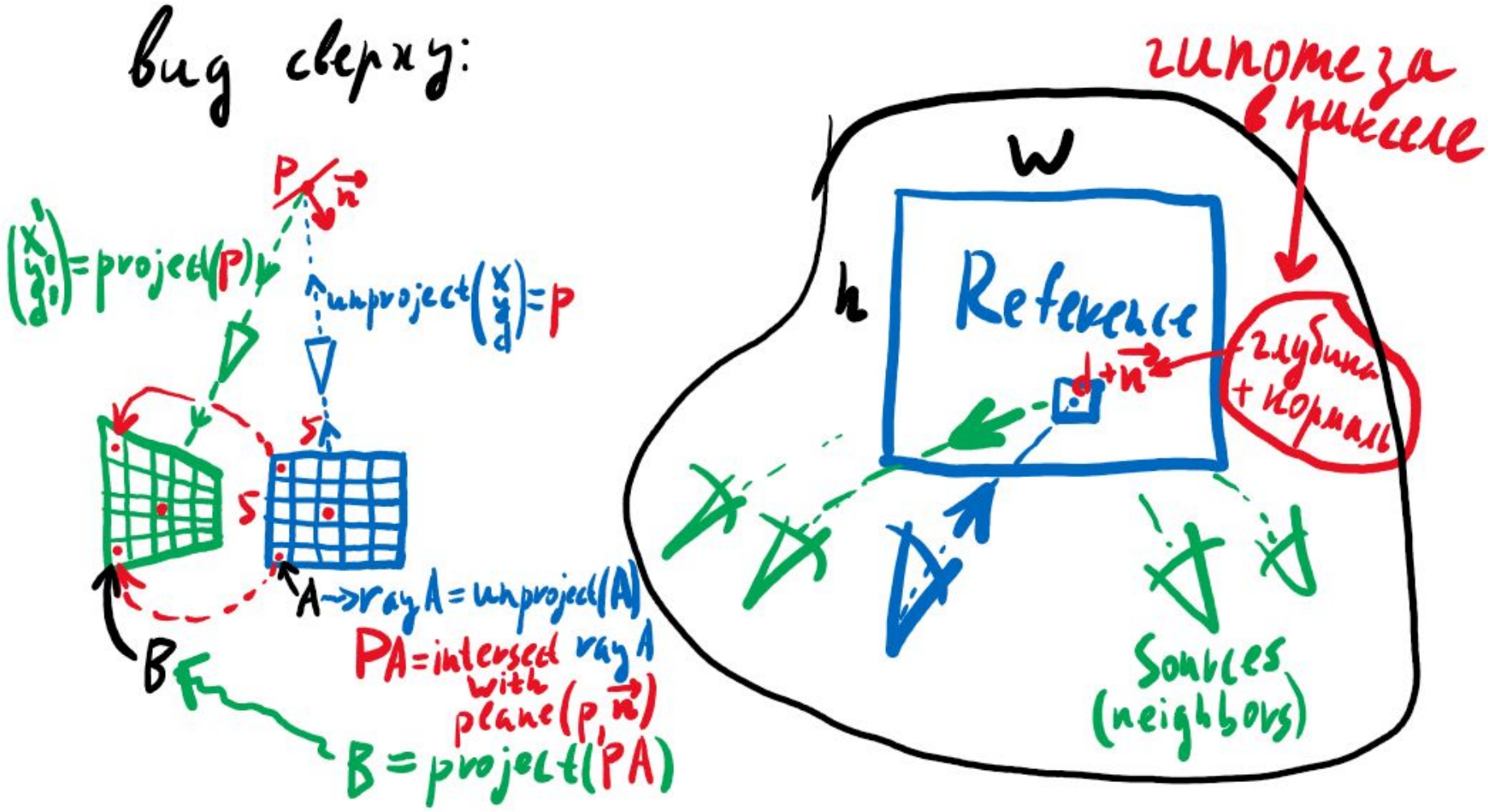
2) Надо оптимизировать не только глубины, но и нормали!

3) Нужна доп. информация - например третья картинка (или еще больше)!

Давайте напрямую оптимизировать 3D пространство!



Как оценить качество гипотезы - **Cost**?



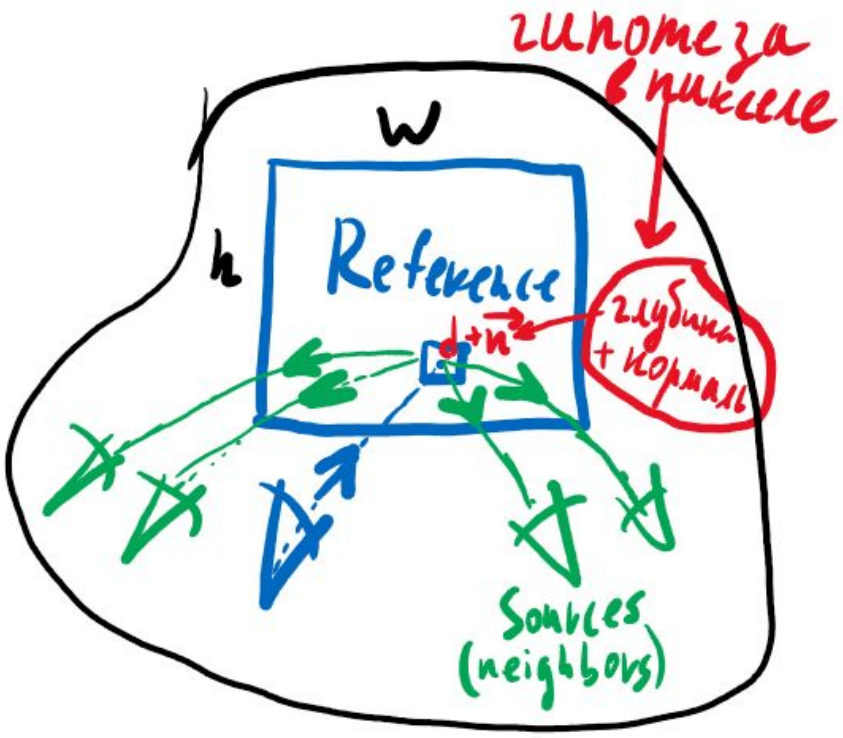

A
B
B
G
D
E


A
B
C
D
E

Как комбинировать **Costs** по нескольким соседям?

$$Cost_i(x, y, d, \vec{n}) = \frac{\sum_{j \in N_i} cost_{ij}(x, y, d, \vec{n})}{|N_i|} \; ?$$

А что если в части кадров эта поверхность заслонена? (**occlusion**)



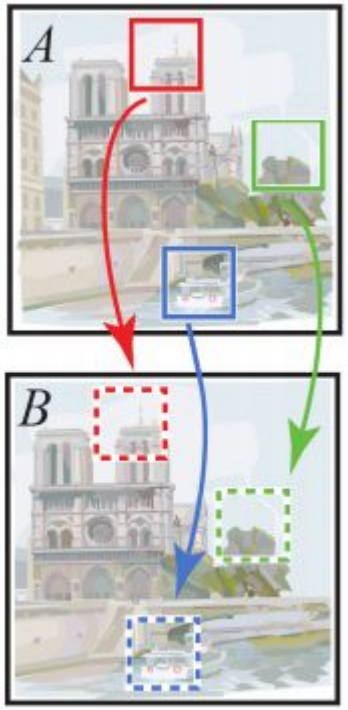
Как искать глубины и нормали?

Полный перебор?

Случайные варианты?

1. PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing

Хотим для каждого патча в A найти самый похожий патч в B.



Natural structure of images: правильный ответ обычно содержит большие связные регионы сопоставления. Можем увеличить эффективность выполняя поиск зависимо от соседей. Т.е. вдоховляясь 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. PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing

1) Начинаем со случайной гипотезы в каждом пикселе

2) Итерируемся:

Propagation:
эксплуатируем согласованность т.е. распространяем хорошие гипотезы по соседям

Random search (Refinement):
текущая гипотеза пробует несколько (разной длины) сдвигов, оставляется лучшее по **Cost** сопоставление

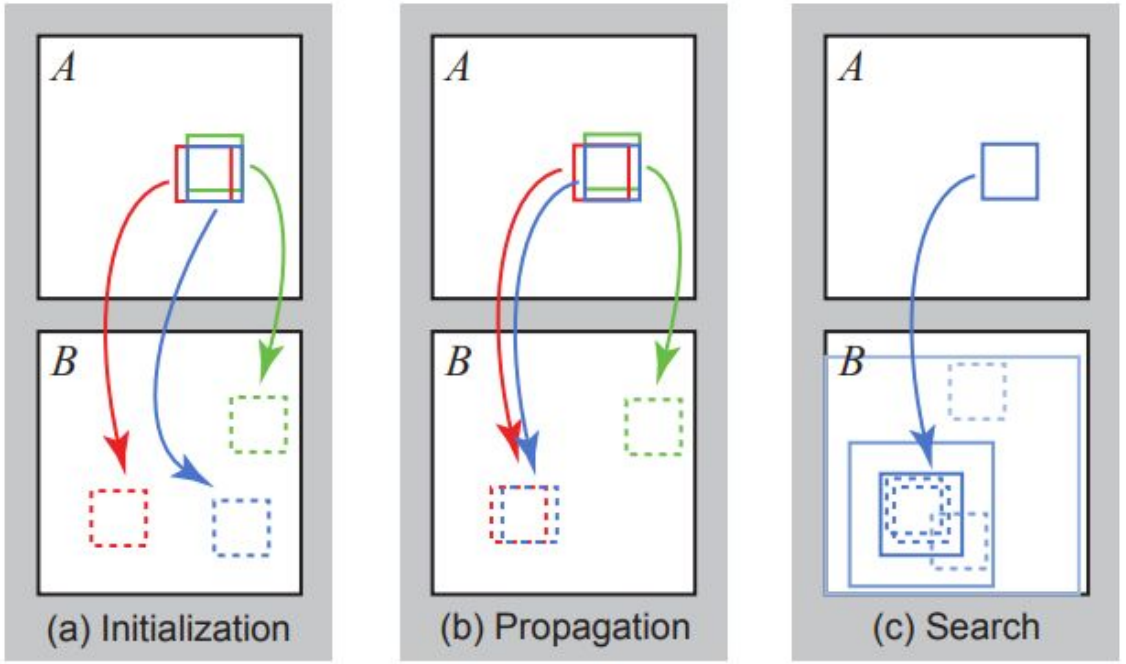


Figure 2: *Phases of the randomized nearest neighbor algorithm: (a) patches initially have random assignments; (b) the blue patch checks above/green and left/red neighbors to see if they will improve the blue mapping, propagating good matches; (c) the patch searches randomly for improvements in concentric neighborhoods.*

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

1. PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing

1) **Инициализация:**
 - Случайные смещения
 - Или **upscale** с результата работы на предыдущей ступени пирамиды детализации (если **coarse-to-fine** схема)
 - Если оставить только **upscale** - останемся в локальном минимуме, поэтому надо несколько раундов пытаться улучшить **Cost** случайными равномерно распределенными гипотезами

2) **Итерируемся:** (строка за строкой сверху вниз, в строке слева направо)
 - **Propagation:** пытаемся улучшить наше сопоставление **f(x, y)** через сопоставление соседа слева **f(x−1, y)** и сопоставление соседа сверху **f(x, y−1)**. На четных итерациях - в обратном порядке идем.
 - **Random search (Refinement):** пробуем улучшить наше сопоставление **f(x, y)** проверяя последовательность случайных кандидатов на экспоненциально уменьшающемся расстоянии от **f(x, y)**

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

1. PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing

(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)

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

Как искать карту глубины?

Полный перебор всех глубин и нормалей в каждом пикселе?

Случайные варианты?

Как искать глубины и нормали?

1) Заполнили карту глубины случайными **глубинами + нормалями**
2) **Propagation:** скан-линией проходим, пытаемся улучшить гипотезу пикселя смотря на соседа сверху и на соседа слева (а на четных итерациях - наоборот).
3) **Refinement:** пытаемся улучшить гипотезу:
   - случайными независимыми кандидатами
   - случайными изменениями текущей гипотезы (**perturbation**)

Как искать глубины и нормали?

1) Заполнили карту глубины случайными **глубинами + нормалями**
2) **Propagation:** скан-линией проходим, пытаемся улучшить гипотезу пикселя смотря на соседа сверху и на соседа слева (а на четных итерациях - наоборот).
3) **Refinement:** пытаемся улучшить гипотезу:
   - случайными независимыми кандидатами
   - случайными изменениями текущей гипотезы (**perturbation**)

Как ускорить сходимость?

Что делать если поверхность снята с разной детализацией (разный масштаб)?

Как пофильтровать ошибки/occlusions? Left-right check?

Как влезть в память если картинок много?

Откуда взять массовый параллелизм для GPU?

Как добавить субпиксельную точность?

2. Gipuma

Gipuma: Massively Parallel Multiview Stereopsis by Surface Normal Diffusion, Galliani et. al., 2015

3. Colmap

Colmap: Pixelwise View Selection for Unstructured Multi-View Stereo, Schonberger et. al., 2016

4. ACMH/ACMM

ACMH/ACMM: Multi-Scale Geometric Consistency Guided Multi-View Stereo, Xu Q, Tao W, 2019

5. Наш PatchMatch

PatchMatch. Ссылки

Benchmarks (удобно находить новые хорошие статьи):

- **ETH3D**: High-resolution multi-view benchmark
- **Tanks And Temples** Benchmark

Методы (удобно находить новые хорошие статьи через цитирования уже известных хороших):

- **Gipuma**: Massively Parallel Multiview Stereopsis by Surface Normal Diffusion, Galliani et. al., 2015
- **Colmap**: Pixelwise View Selection for Unstructured Multi-View Stereo, Schonberger et. al., 2016
- **ACMH/ACMM**: Multi-Scale Geometric Consistency Guided Multi-View Stereo, Xu Q, Tao W, 2019

Исходные коды (**осторожно с лицензиями!**):

- **ACMH/ACMM**: оригинальные ACMH (MIT), ACMM (MIT),
  сторонняя реализация AMCH-ACMM (MIT)
  сторонняя реализация в OpenMVS / ACPMH (**AGPL**)
- **Colmap:** Colmap (BSD)
- **Gipuma:** Gipuma (**GPL**)
- **OpenMVS:** OpenMVS / PatchMatch (**AGPL**)

Разное:

- PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing, Barnes et. al., 2009
- Записки с обзором методов и идеями

Вопросы?

Полярный Николай
polarnick239@gmail.com